Introduction to machine learning

Maksim Kretov

Lecture 6: Deep learning: overview and motivation

5vision, 2017

Course information

Course

10 lectures + 2 seminars; February-May 2017.

Schedule and up-to-date syllabus

https://goo.gl/xExEuL

Contact information and discussion

Maksim Kretov (kretovmk@gmail.com)

Slack group: https://miptmlcourse.slack.com

to get an invite, send e-mail to kretovmk@gmail.com.

Plan of the course

Math and basics of ML	(1-2)			*Theoretical tasks*
Some of ML methods	(3)			
Seminar on ML basics	(4)			
Basics of neural networks	(5)			*+Practical tasks*
<u>Deep learning overview</u>	(6)	← **Today**		
Training deep networks	(7)			
DL for Computer Vision	(8-9)		**Solving more complex ML tasks using NNs**	
DL for time series prediction	(10-11)			
Concluding seminar	(12)			

Plan for the lecture

A. Previous lecture

1. ML tasks
2. ERM framework
3. Backpropagation

B. Deep learning

1. Historical notes
2. Motivation:
3. Success stories

C. Problems in training deep networks

D. Practical assignment

A.1 Previous lectures: ML tasks

Supervised learning:

Training set: $\boldsymbol{D} = \{(\mathbf{x}_{\boldsymbol{n}}, y_n), n = 1, .. N\}$ *(inputs and labels!)*

Y are class ids or numbers => classification or regression

Task to solve: Predict y^* for new input $\mathbf{x}^*$

=> **Focus on accurate prediction**

Unsupervised learning:

Training set: $D = \{\mathbf{x}_{\boldsymbol{n}}, n = 1, .. N\}$ *(just inputs!)*

Task to solve:

Finding compact description of data

A.2 Previous lectures: ERM framework

Empirical risk minimization approach (ERM)

Formula for fitting the model within ERM framework:

$$\theta^{opt} = \mathbf{argmin}_\theta \frac{1}{N}\sum_{n=1}^{N} L(y_n, \hat{y}_n) + \lambda\Omega(\theta)$$

$L\big(y_n, f(\mathbf{x}_n, \theta)\big)$ is loss function; $\hat{y}_n = f(\mathbf{x}_n, \theta)$ is prediction

$\Omega(\theta)$ is a regularizer **=> learning converted into optimization task!**

Training neural networks with ERM. Probabilistic interpretation:

Maximizing likelihood of correct class in predicted distribution.

A.3 Previous lecture: Backpropagation

Calculating gradient w.r.t. parameters

Given: input $\mathbf{x}$

Forward pass:

$$z^1 = \omega^1 \mathbf{x} + b^1$$
$$a^1 = \sigma(z^1)$$
$$z^l = \omega^l a^{l-1} + b^l$$
$$a^l = \sigma(z^l)$$

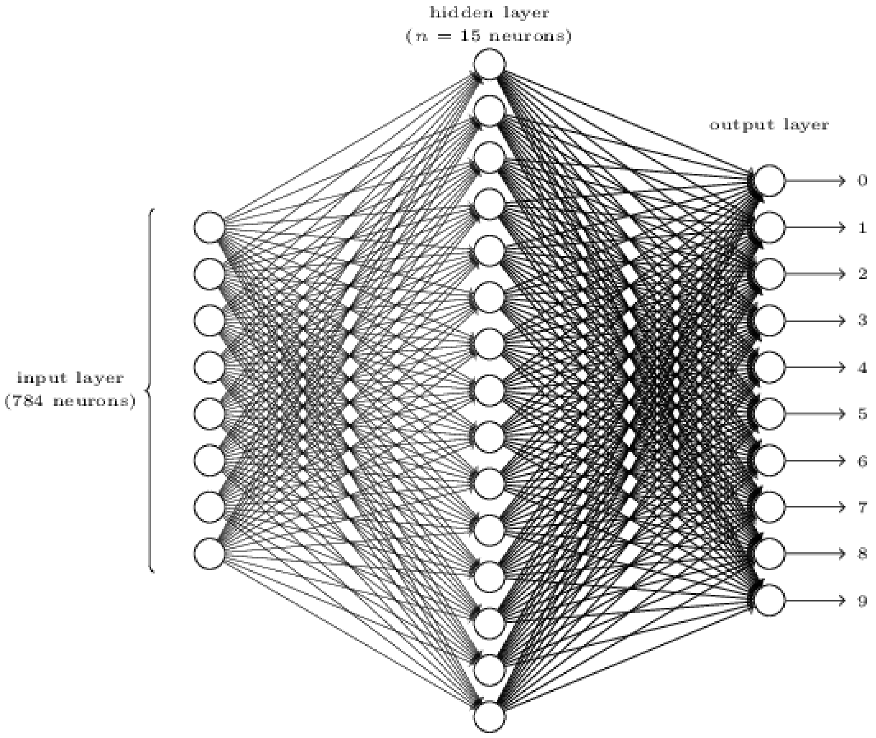
Backward pass:

$$\delta^L = \frac{\partial L}{\partial a^L} \circ \sigma'(z^L)$$
$$\delta^l = \left(\left(\omega^{l+1}\right)^T \delta^{l+1} \right) \circ \sigma'(z^l)$$

Calculating derivatives:

$\frac{\partial L}{\partial b^l} = \delta^l$ *biases*

$\frac{\partial L}{\partial \omega^l} = a^{l-1}\delta^l$ *weights*



* Image from http://neuralnetworksanddeeplearning.com/chap1.html

B.1 Deep learning: Historical overview

70s: Understanding limited power of simple neural models

Shift of research focus onto "symbolic AI"

80s: Great enthusiasm about NNs

Further development of BP algorithm (G Hinton, 1986), Hopfield network, Boltzmann machines, Autoencoders, RBF networks, CNNs

90s: Diversion of focus from NNs to SVMs

SVM with kernel trick (V. Vapnik, 1992)

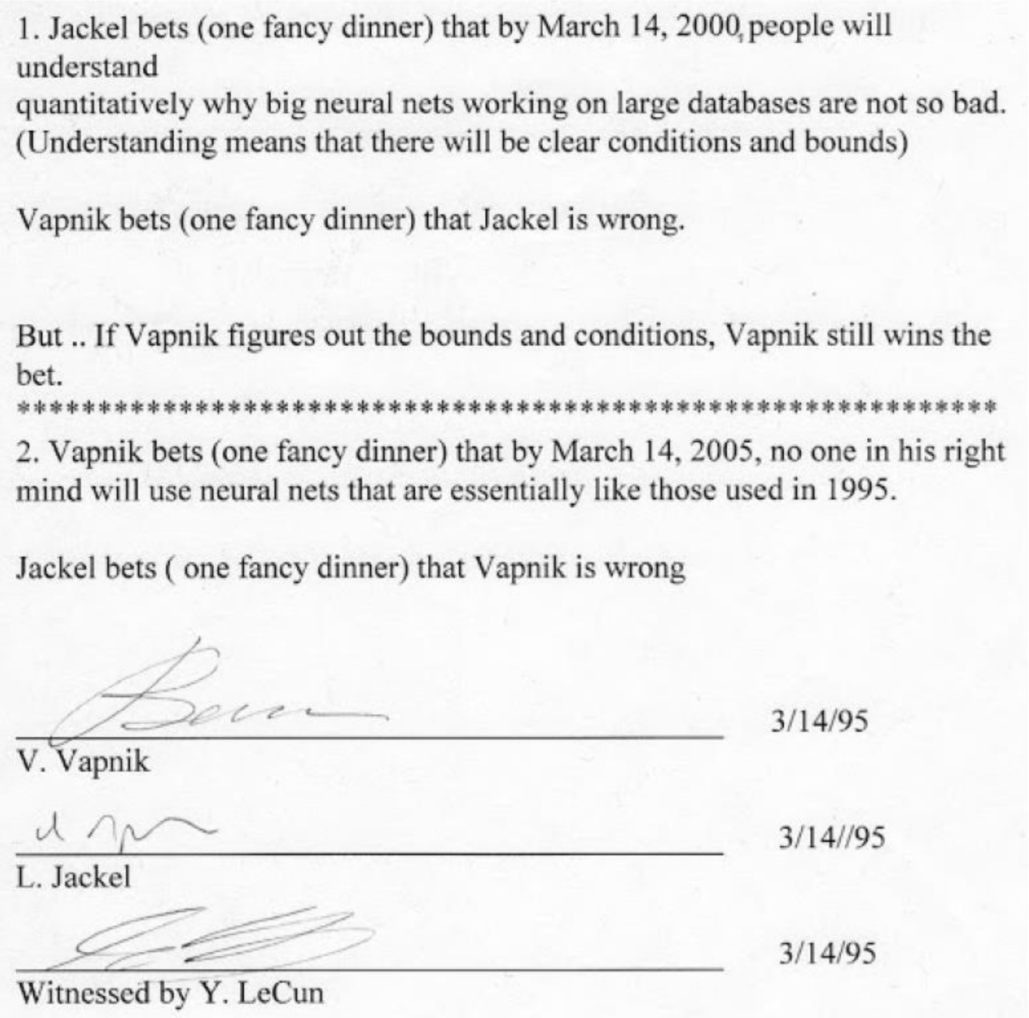
00s: Renewed interest in deep NNs

Eff. training techniques for deep networks (G Hinton, 2006)

10s: Deep learning hype ← **We enjoy it in 2017**

Deep networks beat state-of-the-art (SOTA) results in many areas.

B.1 Deep learning: Historical overview



1. Jackel bets (one fancy dinner) that by March 14, 2000, people will understand
quantitatively why big neural nets working on large databases are not so bad.
(Understanding means that there will be clear conditions and bounds)

Vapnik bets (one fancy dinner) that Jackel is wrong.

But .. If Vapnik figures out the bounds and conditions, Vapnik still wins the bet.

****

2. Vapnik bets (one fancy dinner) that by March 14, 2005, no one in his right mind will use neural nets that are essentially like those used in 1995.

Jackel bets (one fancy dinner) that Vapnik is wrong

V. Vapnik 3/14/95

L. Jackel 3/14//95

Witnessed by Y. LeCun 3/14/95

Deep learning renaissance started in 2006.

In 10s DL techniques achieved SOTA results in image recognition, speech recognition.

BUT:
It is still active area of research to explain "quantitatively" why big neural nets work.

B.2 Deep learning: Motivation

Just an informal definition of “deep” networks:

Networks with up to 3 (2 hidden) layers → shallow

More than 3 layers → deep

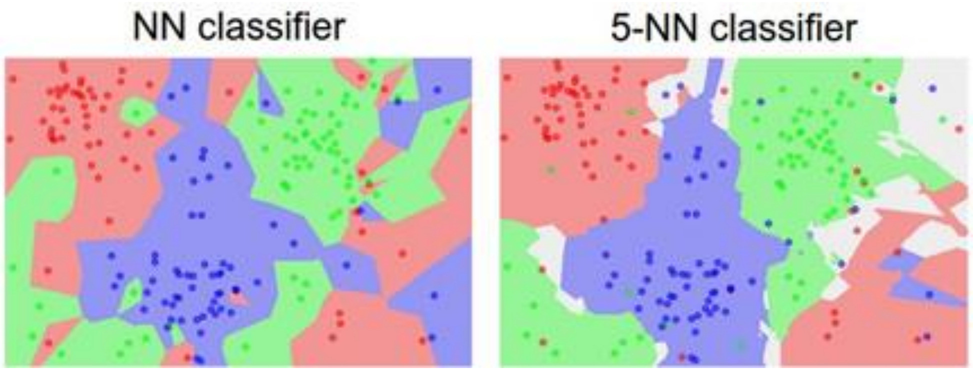
Do we really need these deep neural networks?

According to universal approximation theorem, one hidden layer should be enough.

Let’s consider k-nearest neighbors rule

Universally consistent and *“nearly”* optimal under certain conditions, including number of items in training dataset → ∞ *

Does kNN solve all ML problems? No!



* See thorough discussion in [7]

B.2 Deep learning: Motivation

Curse of dimensionality

Traditional methods use smoothness prior.

Number of possible distinct configurations of a set of variables increases exponentially as the number of variables increases

=> Number of possible configurations of x is much larger than the number of training examples (we simply cannot have this much data).

So, we need (implicitly) introduce another general prior

Now consider how deep learning encode this prior

*Image from [2]

B.2 Deep learning: Motivation

Traditional/Simple methods

Flexible enough and there are theoretical guarantees. But as of now they are struggling with complex real world/perception-like tasks.

Unless we inject some **additional knowledge** how world works into them.

What is Deep Learning (DL)?

Machine learning algorithms based on learning multiple levels of representation/abstraction*:

multilayer neural networks (CNN, RNN) ← **Focus here**

multilayer graphical models (deep belief network, deep Boltzmann machine)

* Definition from NIPS 2015 deep learning tutorial

B.2 Deep learning: Motivation

Informal link: inspiration from biology (visual cortex)

Edges → patches → surfaces →
→ objects (hierarchy)
Cortex can be seen as multilayer architecture with 5-10 layers
Last layer – “semantic meaning”.

=> Compositionality allows better generalization to overcome curse of dimensionality

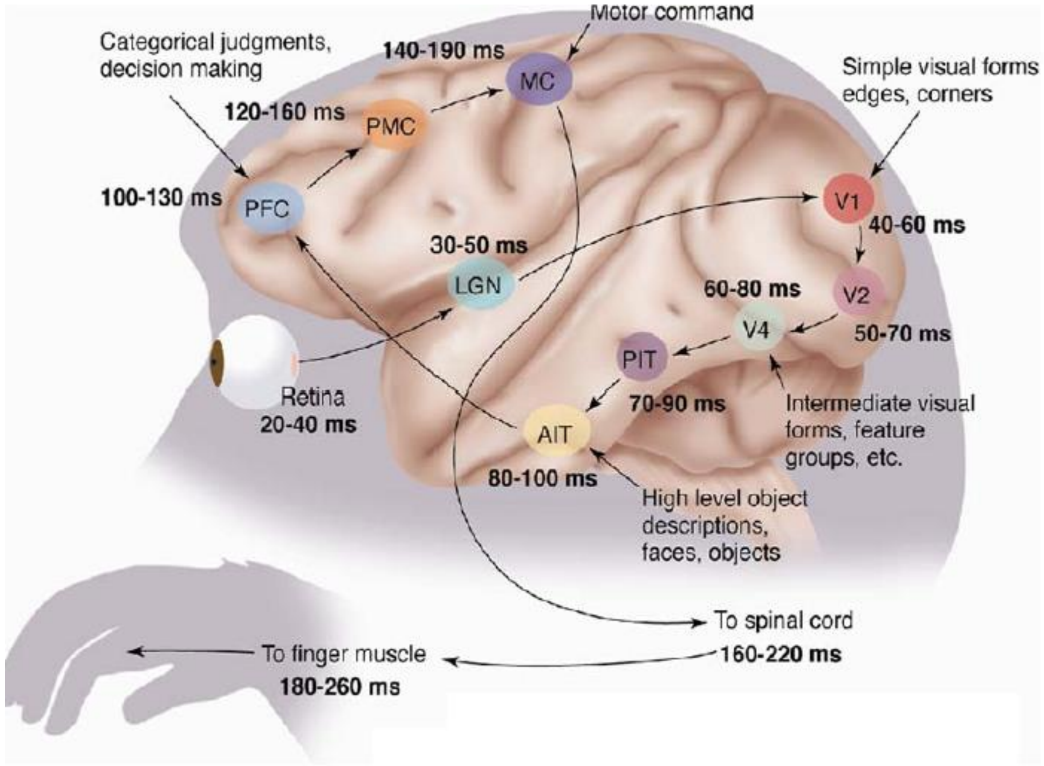
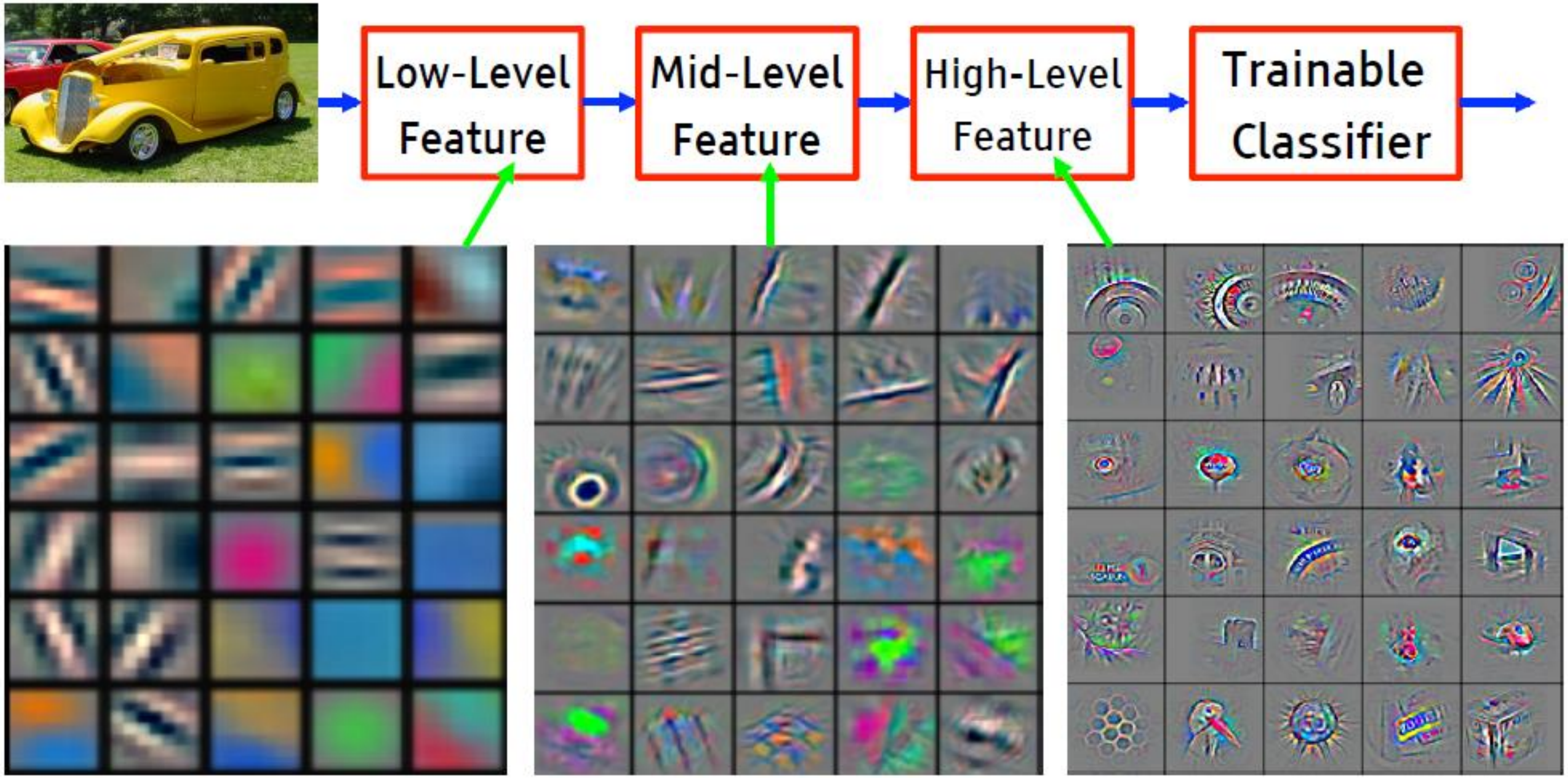


*Image from NIPS 2015 deep learning tutorial

B.2 Deep learning: Motivation



* Image from NIPS 2015 deep learning tutorial

5 minute break..

Questions?

B.2 Deep learning: Motivation

Representation learning

Learning representations of the data that make it easier to extract useful information when building classifiers or other predictors.

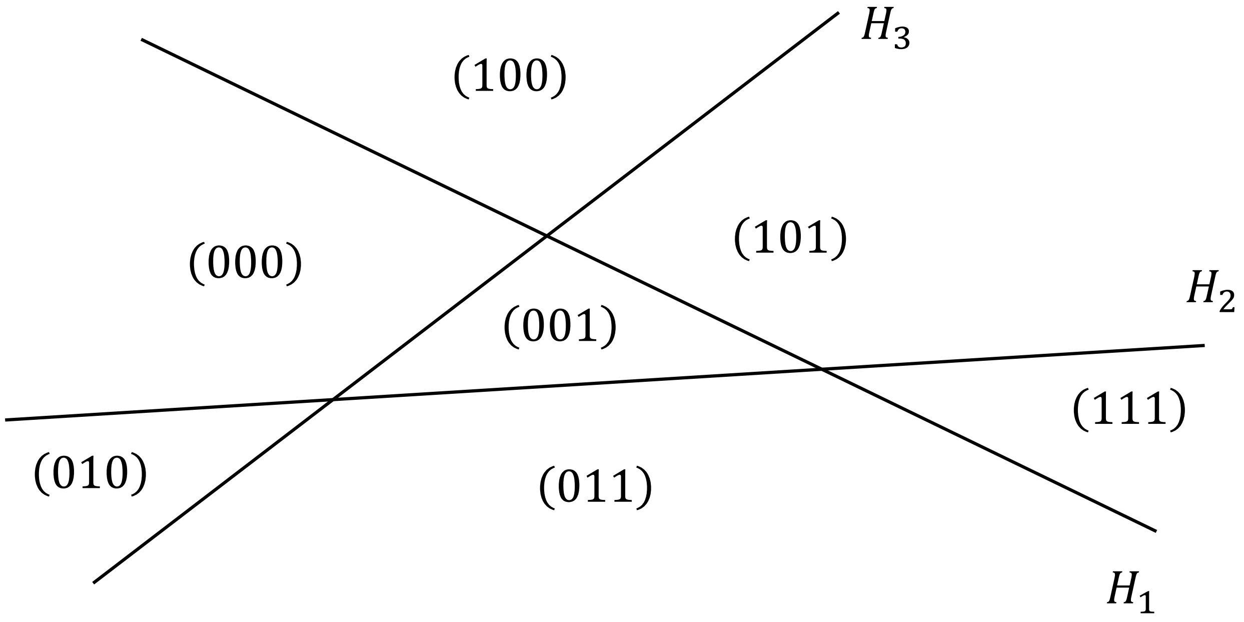
Connection with DL:

DL discovers intricate structure in large data sets using BP algorithm (no feature engineering required).

DL allows to build **compositionality** into ML models => **exponential gain** in representational power.

B.2 Deep learning: Motivation

Distributed representations: Example



7 distinct regions

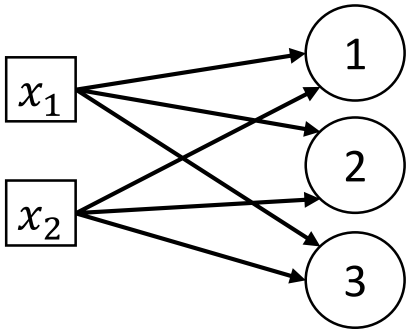
One-hot encoding of each area:

Vector of length 7

Distributed representation:

Vector of length 3

Can be exponentially more efficient.



We have just overcome curse of dimensionality!

Non-local generalization

B.2 Deep learning: Motivation

Each layer corresponds to distributed representation

Good representations are expressive: representation is of reasonable size and can capture many input configurations ("distributed representations").

NOTE: **linear classifier on top of the distributed representation is not able to assign different class identities to every neighboring region**

1. **Learners of one-hot representations** (local methods): kNN, decision trees

Require $O(N)$ parameters to distinguish N input regions.

2. **Learners of distributed or sparse representations**: RBM, auto-encoders

Can represent $O\left(2^k\right)$ input regions using only $O(N)$ parameters (k is the number of non-zero elements in representation).

Features are not mutually exclusive, and some attributes may be shared between different classes.

* See thorough discussion in [2] Ch 15.

B.2 Deep learning: Motivation

Ok, distributed representations are good

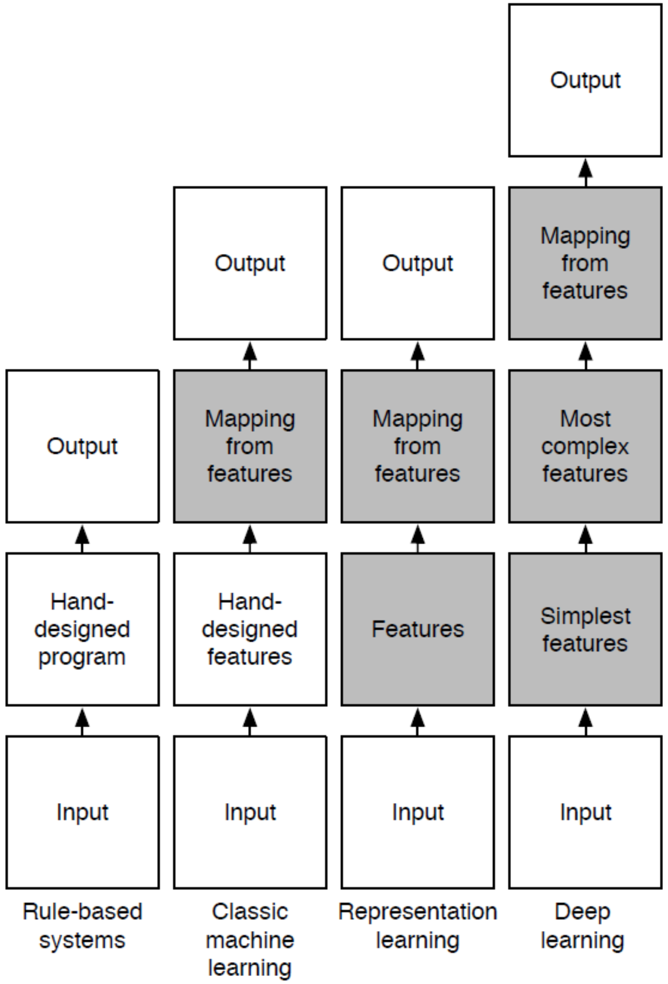
And what about depth?

Function which can compactly be realized by *k* layers of logic elements, may need an exponentially large number of elements if it is realized via *k* − 1 layers.

Example

Parity function with l inputs requires $O(l^2)$ parameters for a neural network with one hidden layer, $O(l)$ parameters and nodes for a multilayer network with $O(\log_2 l)$ layers [2].

B.2 Deep learning: Motivation



It only works because we are making some assumptions about the data generating distribution (reminder: **no free lunch theorem**).

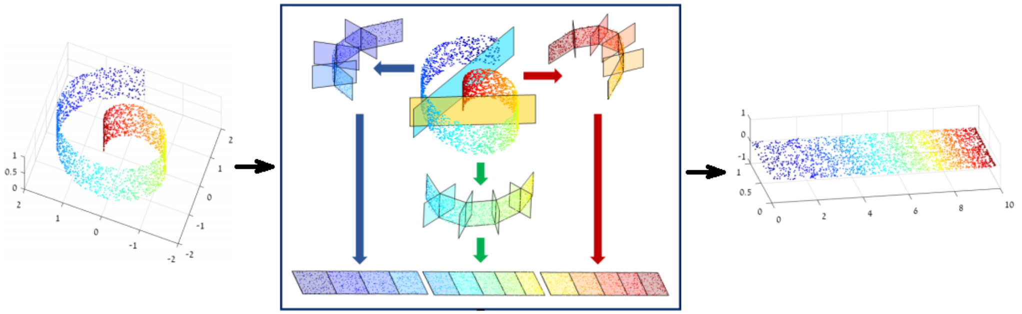
Worse-case distributions still require exponential data.

Data was generated by the **composition of factors** or features, potentially at multiple levels in a hierarchy**.

* Image from NIPS 2015 deep learning tutorial, **See thorough discussion in [2]

B.2 Deep learning: Motivation

After 2014: Articles on other interpretations occurred.



1. DL and manifold hypothesis

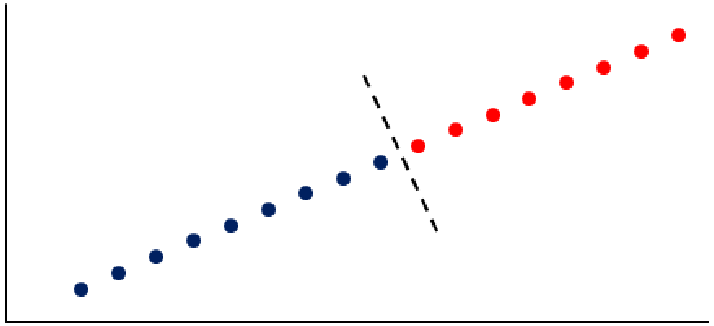
Learning manifold =>

=> coordinate system on low-dim manifold =>

=> rewrite input vectors in new coords => classifier

2. Physical interpretations

Track functions of interest back to laws of physics.



See refs in the end of presentation

*Image from http://arxiv.org/abs/1602.04723

B.3 Deep learning: Success stories

Why now? Deep learning success premises:

We understand learning better

Model structure matters a lot => powerful priors

No need to be scared of non-convex optimization (see next slides)

Exponential growth of computational power (GPUs)

So experiments and tests are cheap

The availability of massive amounts of labeled data

We replaced hand-engineering with knowledge extracted from data

All ingredients are crucial.

B.3 Deep learning: Success stories

Representation learning: examples

1. word2vec

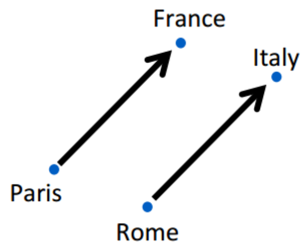
One-hot encoding: $(0 \ldots 010 \ldots 0)$

dim 10,000-100,000

Word embedding: $(x_1, \ldots x_m)$, e.g. $(0.2, \ldots 1.3)$

dim 100-300

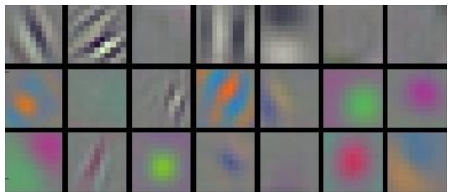
Letters → one-hot vector → word embedding



King – Queen ≈ Man – Woman
Paris – France + Italy ≈ Rome

2. Image classification

Pixels → Lines, geometric figures



* Image from NIPS 2015 deep learning tutorial

B.3 Deep learning: Success stories

Examples of deep learning: Supervised learning

Recurrent networks: NLP, translation

Convolutional networks: Image recognition, speech recognition

Memory networks: dialogue systems

Examples of deep learning: Unsupervised learning

Generative modelling with Neural networks, pre-training

Examples of deep learning: Reinforcement learning

End-to-end architectures: mapping from images to actions

B.3 Deep learning: Success stories

ImageNet: ILSVRC 2012 – Classification Task

Top Rankers

1. SuperVision (**0.153**): Deep Conv. Neural Network (Krizhevsky et al.)
2. ISI (0.262): Features + FV + Linear classifier (Gunji et al.)
3. OXFORD_VGG (0.270): Features + FV + SVM (Simonyan et al.)
4. XRCE/INRIA (0.271): SIFT + FV + PQ + SVM (Perronin et al.)
5. University of Amsterdam (0.300): Color desc. + SVM (van de Sande et al.)

(Krizhevsky et al., 2012)

*Images from cs231n course and NIPS’15 DL tutorial

B.3 Deep learning: Success stories

ImageNet: ILSVRC 2013 – Classification Task

Top Rankers

1. Clarifi (**0.117**): **Deep Convolutional Neural Networks (Zeiler)**
2. NUS: **Deep Convolutional Neural Networks**
3. ZF: **Deep Convolutional Neural Networks**
4. Andrew Howard: **Deep Convolutional Neural Networks**
5. OverFeat: **Deep Convolutional Neural Networks**
6. UvA-Euvision: **Deep Convolutional Neural Networks**
7. Adobe: **Deep Convolutional Neural Networks**
8. VGG: **Deep Convolutional Neural Networks**
9. CognitiveVision: **Deep Convolutional Neural Networks**
10. decaf: **Deep Convolutional Neural Networks**
11. IBM Multimedia Team: **Deep Convolutional Neural Networks**
12. Deep Punx (0.209): **Deep Convolutional Neural Networks**
13. *MIL (0.244): Local image descriptors + FV + linear classifier (Hidaka et al.)*
14. Minerva-MSRA: **Deep Convolutional Neural Networks**
15. Orange: **Deep Convolutional Neural Networks**
16. BUPT-Orange: **Deep Convolutional Neural Networks**
17. Trimps-Soushen1: **Deep Convolutional Neural Networks**
18. *QuantumLeap: 15 features + RVM (Shu&Shu)*

*Images from cs231n course and NIPS'15 DL tutorial

C. Problems in training deep networks

Why training deep networks is hard?

1. First hypothesis: optimization is harder (underfitting)

Vanishing gradient problem

Saturated units block gradient propagation

2. Second hypothesis: overfitting

Space of complex functions

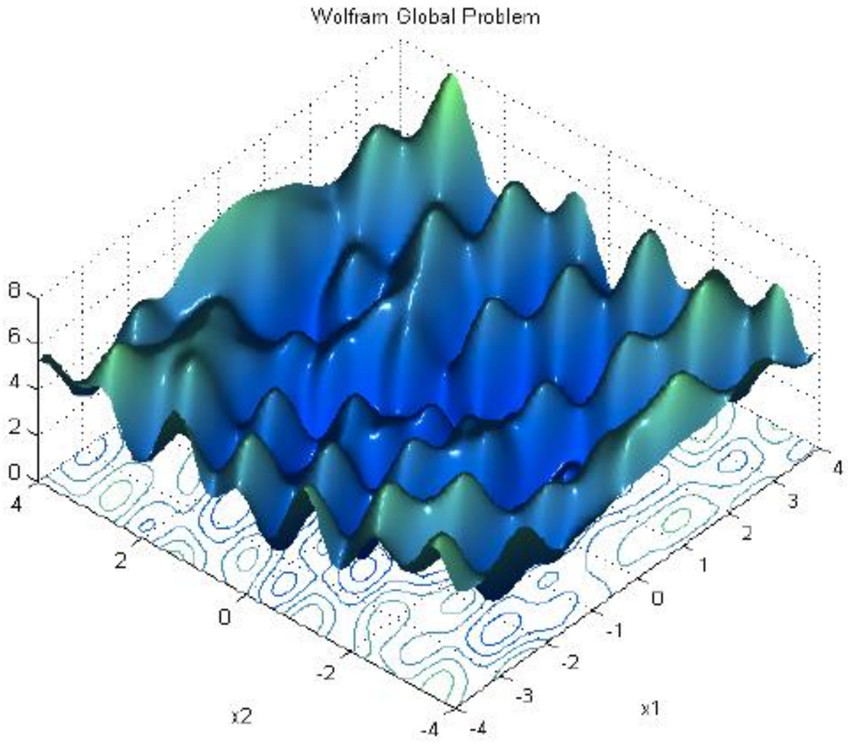
Deep networks usually have lots of parameters

C. Problems in training deep networks

Saddle points vs Local minimums

Local minima dominate in low-D,
but saddle points dominate in high-D

Most local minima are close to the
bottom (global minimum error)



* Image from NIPS 2015 deep learning tutorial

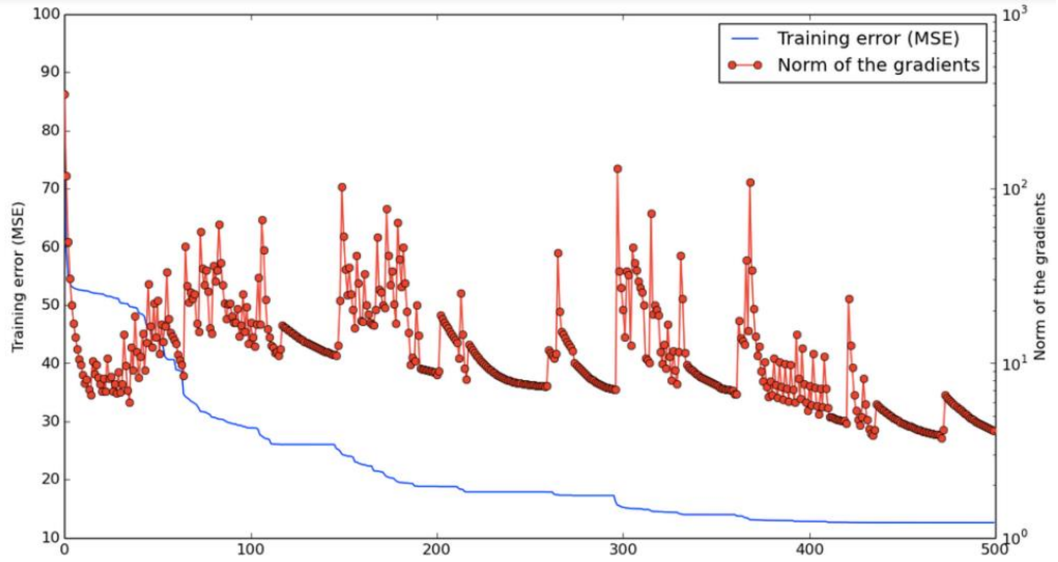
C. Problems in training deep networks

Saddle points during training

Oscillating between two behaviors:

Slowly approaching a saddle point

Escaping it



* Image from NIPS 2015 deep learning tutorial

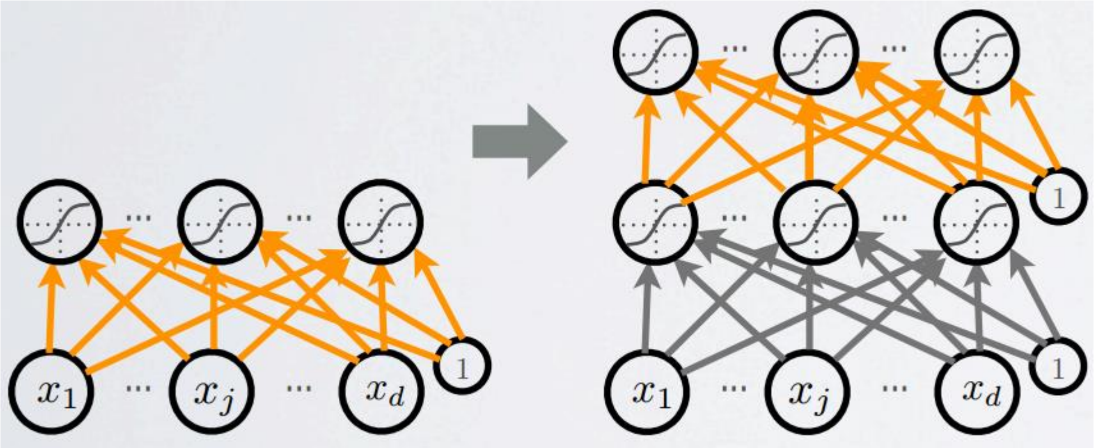
C. Problems in training deep networks

Unsupervised pre-training

Initialize hidden layers using unsupervised learning: force network to represent latent structure of input distribution

character image

random image



Greedy, layer-wise procedure. Once pre-training is done – add output layer and usual BP algorithm.

* Image from Course of H.Larochelle: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

C. Problems in training deep networks

Unsupervised pre-training

first layer: hidden unit features that are more common in training inputs than in random inputs

second layer: combinations of hidden unit features that are more common than random hidden unit features

third layer: combinations of combinations of ...

etc.

C. Problems in training deep networks

Transfer learning

Pre-train a convolutional network on a very large dataset

Algorithm:

Convolutional network as fixed feature extractor

Fine-tuning the convolutional network

Next week

Training Neural Networks: better and faster

Improving convergence of training process

- Weights initialization
- Loss function
- Regularization
- Advanced GD

Different architectures of neural networks

D. Homework

For all

1. Reading Ch.6 in [2].

2. Practical assignment #1. For those who is not going to do whole practical assignment, just do part 3 of it (non-programmatic part).

Reading for enthusiasts (after making basic homework)

https://www.reddit.com/r/MachineLearning/comments/50a2x0/max_tegmark_explains_via_physics_why_deep/

https://arxiv.org/pdf/1608.08225.pdf

https://arxiv.org/pdf/1611.00740.pdf

https://medium.com/intuitionmachine/the-holographic-principle-and-deep-learning-52c2d6da8d9

https://arxiv.org/pdf/1402.1869.pdf

Refs

1. Thorough review of relevant math topics:
http://info.usherbrooke.ca/hlarochelle/ift725/review.pdf

2*. Ian Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning.

3. Kevin P. Murphy, Machine Learning: A probabilistic perspective.

4. David Barber, Bayesian Reasoning and Machine Learning.

5. Sergios Theodoridis, Machine Learning: A Bayesian and optimization perspective.

6*. See also refs in practical assignment and online courses, especially one from Hugo Larochelle (presentation from lecture 1).

7. L. Devroye, L. Gyorfi, G.A. Lugosi, A Probabilistic Theory of Pattern Recognition, Springer Verlag city, 1996.