

Machine Learning

Neural Networks

Carl Henrik Ek - carlhenrik.ek@bristol.ac.uk

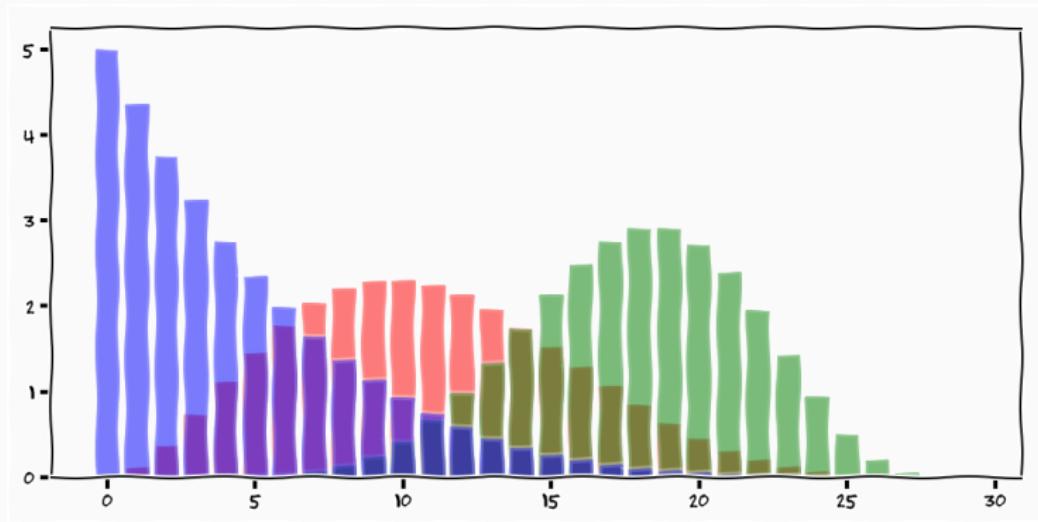
November 11, 2019

<http://www.carlhenrik.com>

Latent Dirichlet Allocation

Text Data

Bag-of-Words



Topic Data

Text collection of words

Images collection of visual words

Genomics collection of sequences

Bioinformatics collection of sequences

Anything that we can describe reliably as a random collection of things fits well into this modelling paradigm

Assumptions

- Documents contains a blend of topics
- Each topic has a characteristic distribution of words
- For simplicity lets assume a finite (known) number of words

Topic Distribution

Topic	$p(\text{Topic})$
Computer Science	0.3
Pugs	0.5
Bristol City	0.1
C64	0.1

$$p(\text{Topic}) = p(\beta)$$

Word Topic Distribution

Topic	8-bit	Walsh	AI	Hugs
Computer Science	0.6	0.09	0.3	0.01
Pugs	0.1	0.1	0.3	0.5
Bristol City	0.1	0.8	0.05	0.05
C64	0.8	0.01	0.15	0.04

$$p(\text{Word}|\text{Topic}) = p(w_{d,n}|\beta_k)$$

Document Topic Distribution

Document	Computer Science	Pugs	Bristol City	C64
One Team in Bristol	0.1	0.2	0.69	0.01
Coursework report	0.8	0.1	0.07	0.03
Dr. Doobs	0.6	0.1	0.1	0.2
Pug Weekly	0.1	0.8	0.05	0.05

$$p(\theta_d)$$

Topic Model

$w_{d,n}$ n :th word in d :th document

Topic Model

$w_{d,n}$ n :th word in d :th document

β_k topic-word distribution for k :th topic

Topic Model

$w_{d,n}$ n :th word in d :th document

β_k topic-word distribution for k :th topic

θ_d topic distribution for d :th document

Topic Model

$w_{d,n}$ n :th word in d :th document

β_k topic-word distribution for k :th topic

θ_d topic distribution for d :th document

$z_{d,n}$ topic assignment for n :th word in d :th document

Topic Model

$w_{d,n}$ n :th word in d :th document

β_k topic-word distribution for k :th topic

θ_d topic distribution for d :th document

$z_{d,n}$ topic assignment for n :th word in d :th document

Now lets write down the joint distribution of these according to our generative model

Topic Model

- To generate the n :th word in the d :th document I need to know the topic that has been assigned to this word and the topic-word distribution

$$p(w_{d,n} | \beta, z_{d,n})$$

Topic Model

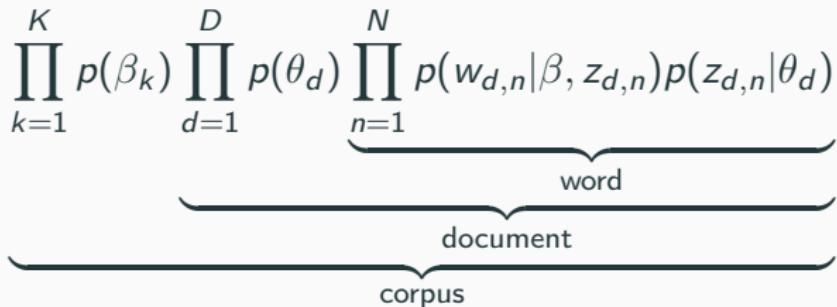
- To generate the n :th word in the d :th document I need to know the topic that has been assigned to this word and the topic-word distribution

$$p(w_{d,n}|\beta, z_{d,n})$$

- To generate a topic assignment $z_{d,n}$ I need to know the topic distribution for document d

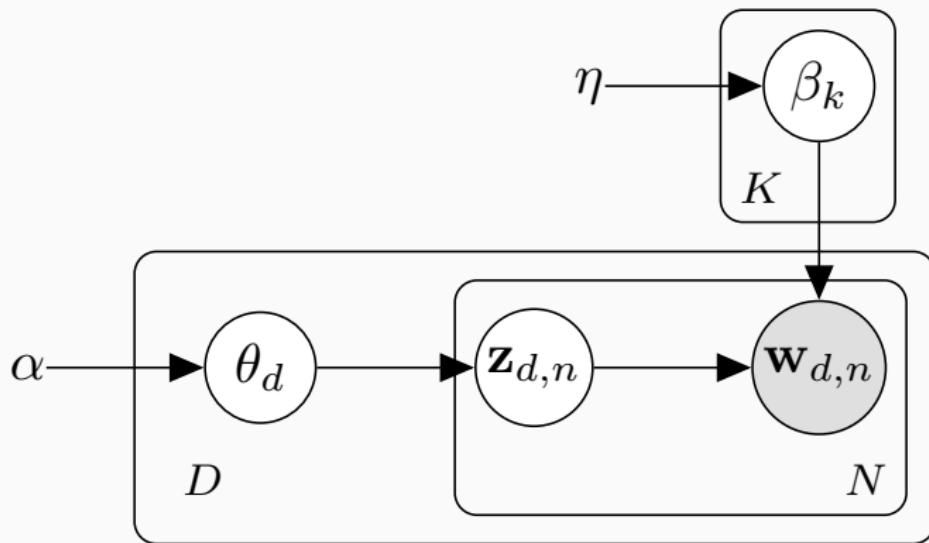
$$p(z_{d,n}|\theta_d)$$

Topic Model

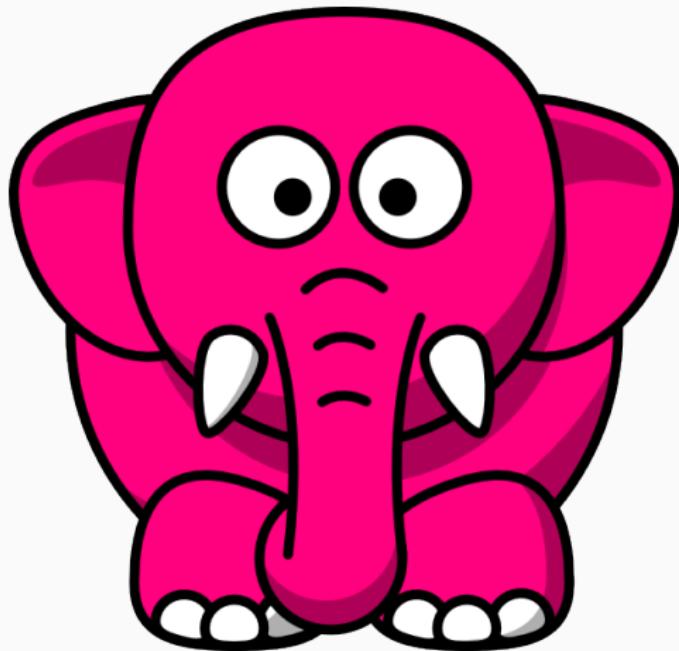
$$p(w, z, \theta, \beta) = \prod_{k=1}^K p(\beta_k) \prod_{d=1}^D p(\theta_d) \prod_{n=1}^N p(w_{d,n} | \beta, z_{d,n}) p(z_{d,n} | \theta_d)$$


- This is the joint distribution of a specific topic model
- The assumptions we have made are called a Latent Dirichlet Allocation [1]
- This specifies a specific probability distribution by our assumptions

Graphical Model



Introduction



Darthmouth workshop¹



¹https://en.wikipedia.org/wiki/Dartmouth_workshop

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

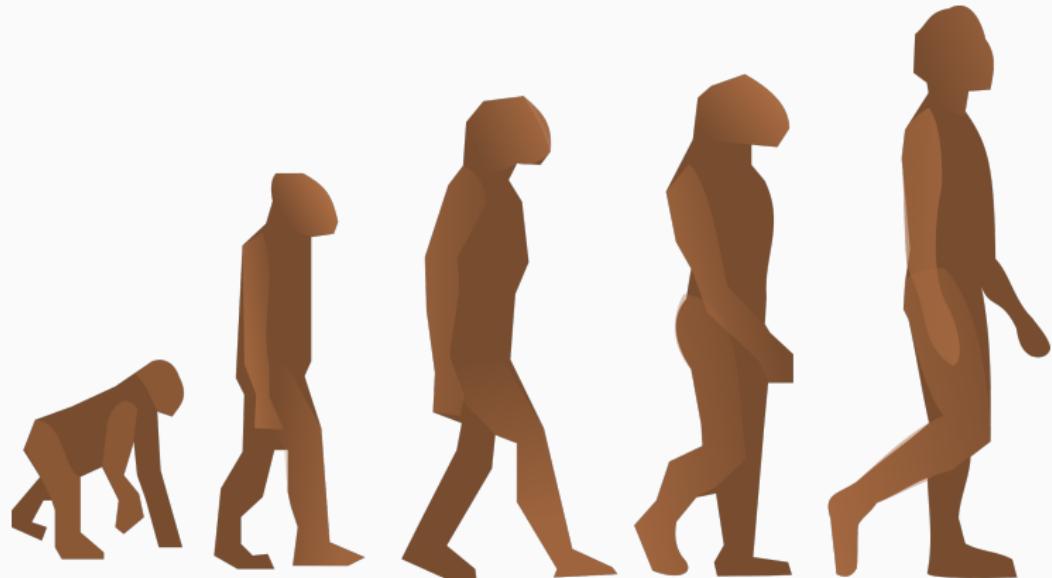
– August 31st 1955

- Artificial Intelligence
 - the idea that intelligence could be described as a set of logical arguments and rules
 - what are the foundational rules of intelligence?

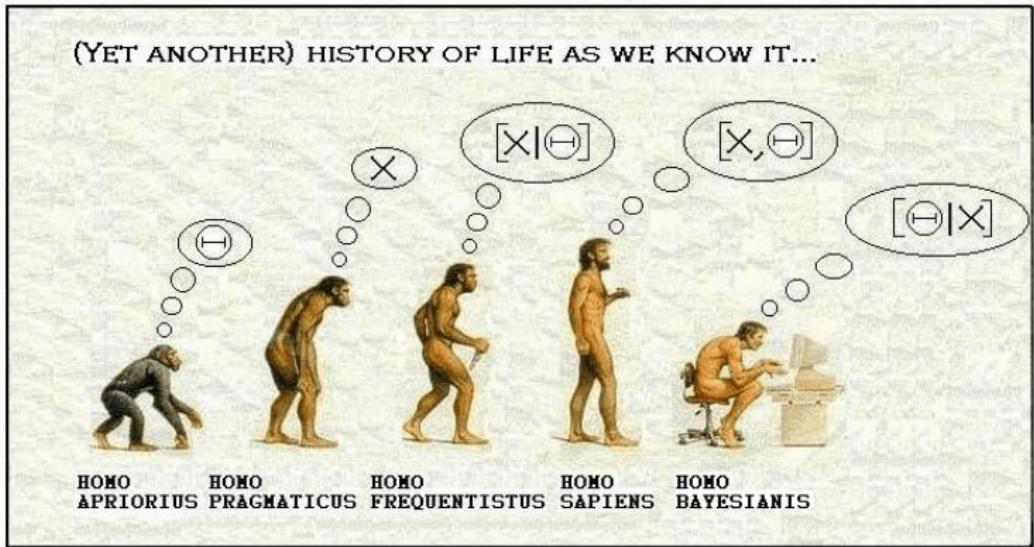


- Artificial Intelligence
 - the idea that intelligence could be described as a set of logical arguments and rules
 - what are the foundational rules of intelligence?
- Machine learning
 - it is naive to think that we can understand everything explicitly (Laplace)
 - can we describe rules for how to learn from evidence/data?

The learning machine



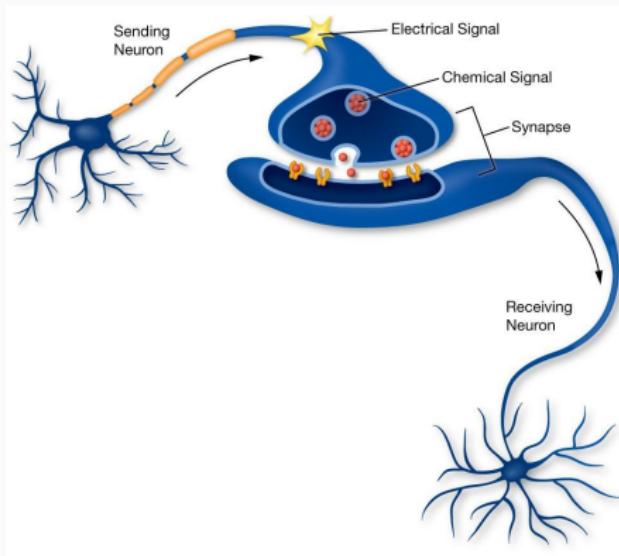
Another view



Carbon Copy



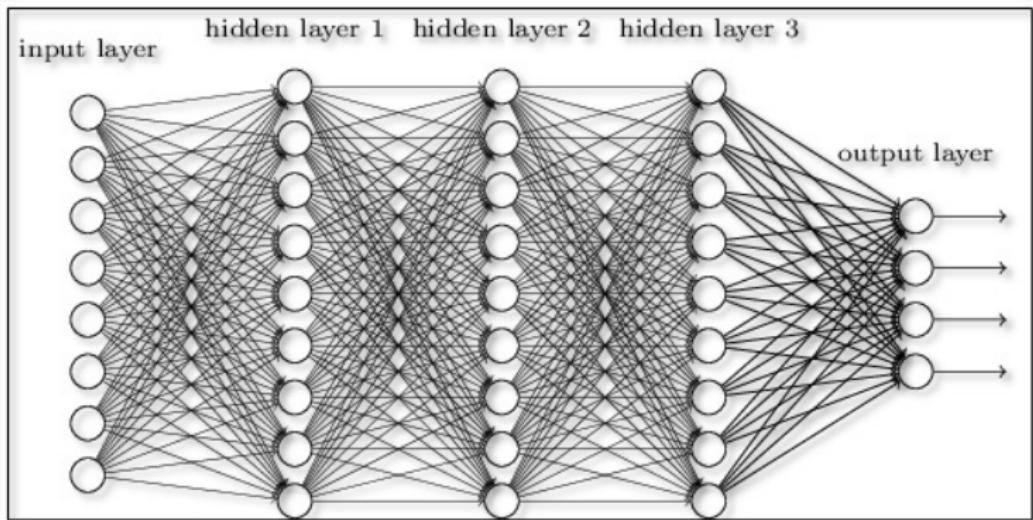
Neurons and spike trains

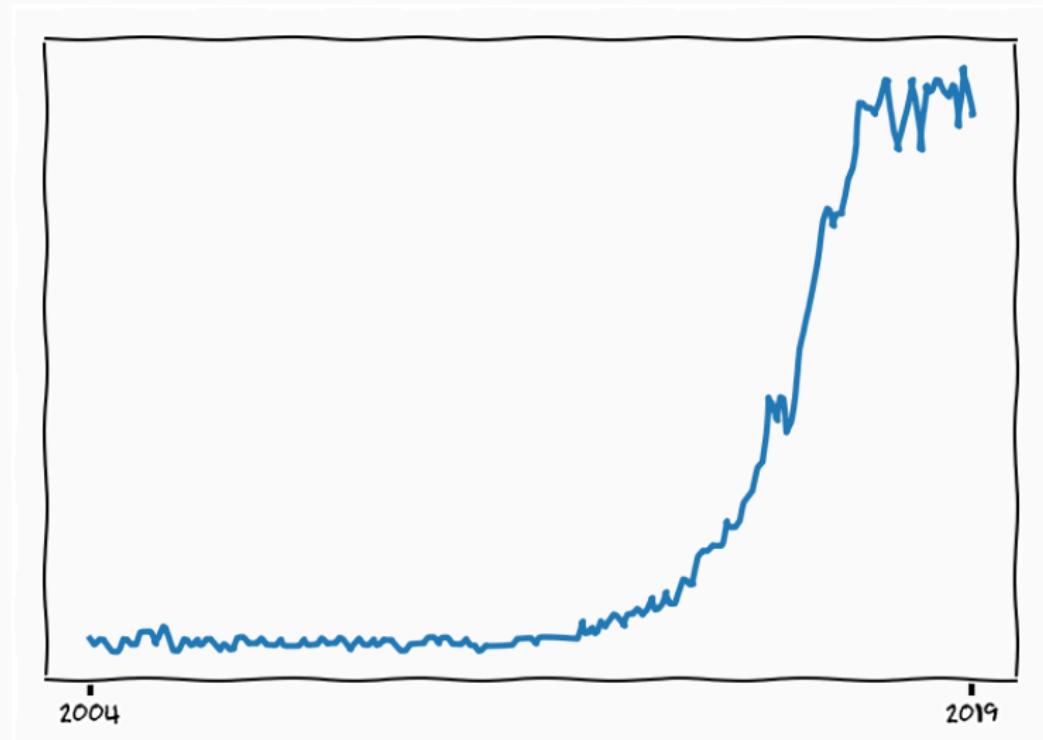


Neural Networks

- First described in 1940 by Walter Pitts
- First computational model called **Perceptron** described in Rosenblatt 1958
- Idea that we should composite a function into several functions
- Usually motivated by "neuroscience"

Neural Networks





Neural Networks

$$y(\mathbf{x}, \mathbf{w}) = f \left(\sum_{j=1}^M w_j \phi_j(\mathbf{x}) \right)$$

- regression: f is identity function
- classification: f is nonlinear function
- f : an activation function

Neural Networks: hidden layer

$$a_j = \sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

$$z_j = h(a_j)$$

a_j activation

w_{ji} weight

w_{j0} bias

z_j hidden units

$h(\cdot)$ activation function

Neural Networks: output layer

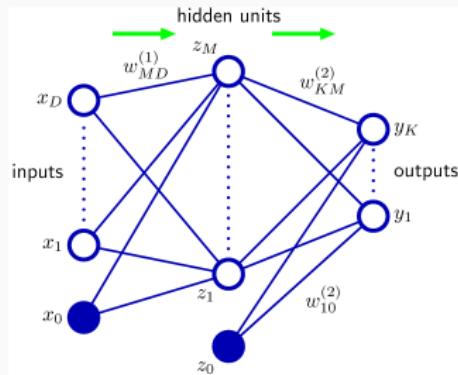
$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

$$y_k = f(a_k) = \sigma(a_k)$$

y_k output variable

$f(\cdot)$ output activation

Neural Network



$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_j^M w_{kj}^{(2)} h \left(\sum_j^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

Composite Functions

Why are composite functions attractive?

$$y = g(x) = f_K(f_{K-1}(f_{K-2}(\dots f_1(x) \dots)))$$

- Kernel of a function

$$\text{Kern}(f_k) = \{(\mathbf{x}, \mathbf{x}') | f_k(\mathbf{x}) = f_k(\mathbf{x}')\}$$

- Kernel of a function

$$\text{Kern}(f_k) = \{(\mathbf{x}, \mathbf{x}') | f_k(\mathbf{x}) = f_k(\mathbf{x}')\}$$

- Image of a function

$$\text{Im}(f_k(\mathbf{x})) = \{\mathbf{y} \in Y | \mathbf{y} = f_k(\mathbf{x}), \mathbf{x} \in X\}$$

- Rank-Nullity Theorem

$$\dim(\text{Im}(f)) = \dim(V) - \dim(\text{Kern}(f))$$

- Rank-Nullity Theorem

$$\dim(\text{Im}(f)) = \dim(V) - \dim(\text{Kern}(f))$$

- Kernel of function

$$\text{Kern}(f_1) \subseteq \text{Kern}(f_{k-1} \circ \dots \circ f_2 \circ f_1) \subseteq \text{Kern}(f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1)$$

- Rank-Nullity Theorem

$$\dim(\text{Im}(f)) = \dim(V) - \dim(\text{Kern}(f))$$

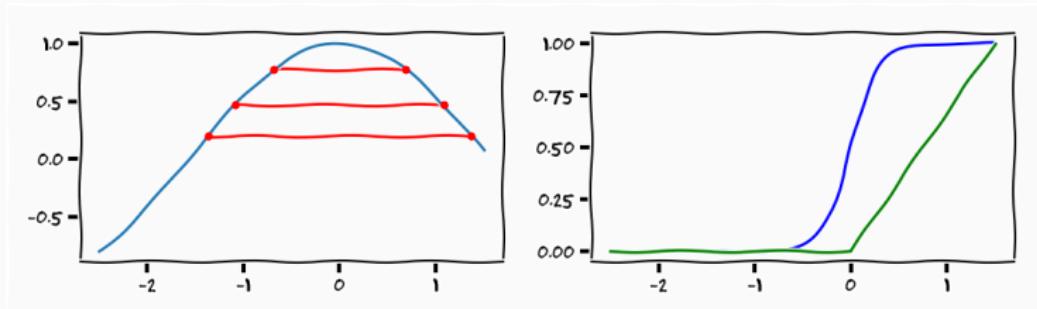
- Kernel of function

$$\text{Kern}(f_1) \subseteq \text{Kern}(f_{k-1} \circ \dots \circ f_2 \circ f_1) \subseteq \text{Kern}(f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1)$$

- Image of a function

$$\text{Im}(f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1) \subseteq \text{Im}(f_k \circ f_{k-1} \circ \dots \circ f_2) \subseteq \dots \subseteq \text{Im}(f_k)$$

Composition functions

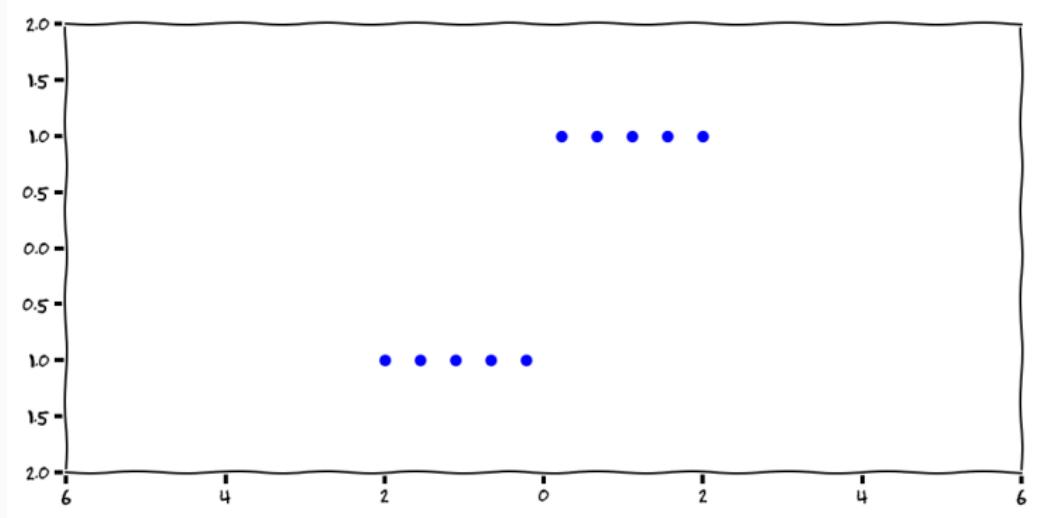


$$y = f_k(f_{k-1}(\dots f_0(x))) = f_k \circ f_{k-1} \circ \dots \circ f_1(x)$$

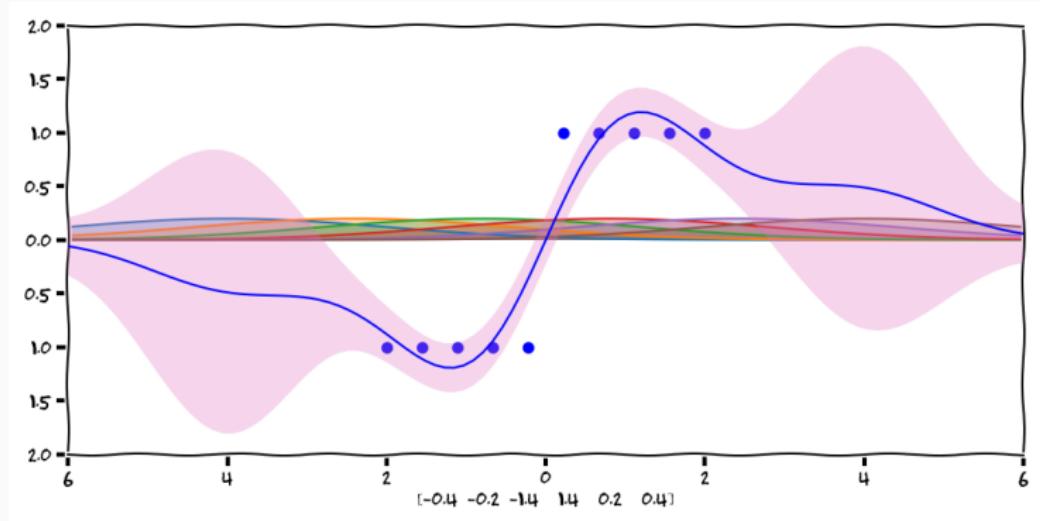
$$\text{Kern}(f_1) \subseteq \text{Kern}(f_{k-1} \circ \dots \circ f_2 \circ f_1) \subseteq \text{Kern}(f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1)$$

$$\text{Im}(f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1) \subseteq \text{Im}(f_k \circ f_{k-1} \circ \dots \circ f_2) \subseteq \dots \subseteq \text{Im}(f_k)$$

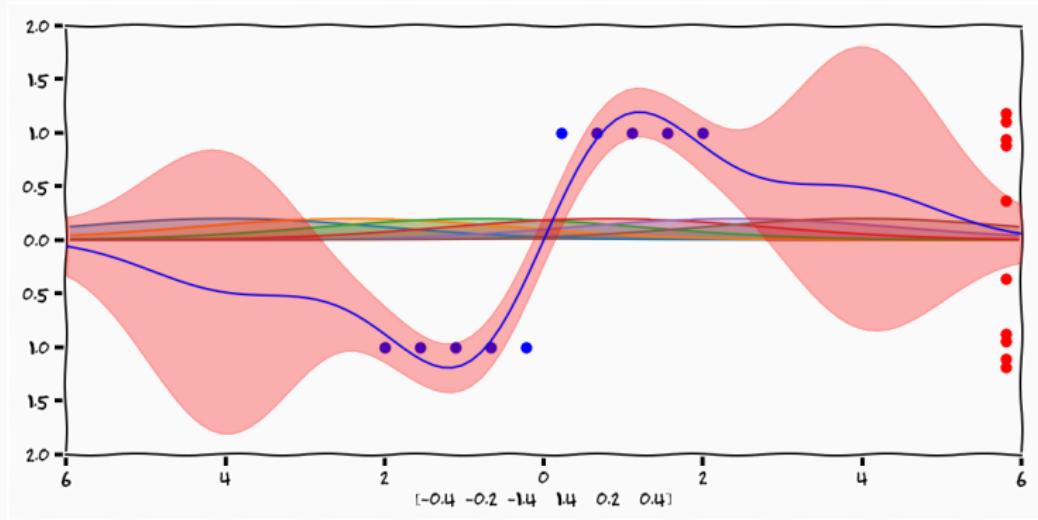
Composite Functions



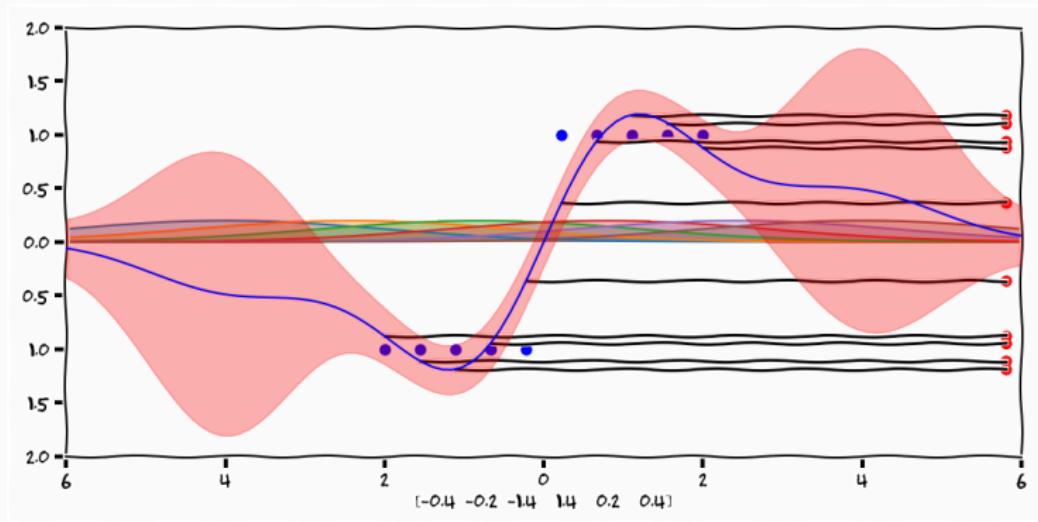
Composite Functions



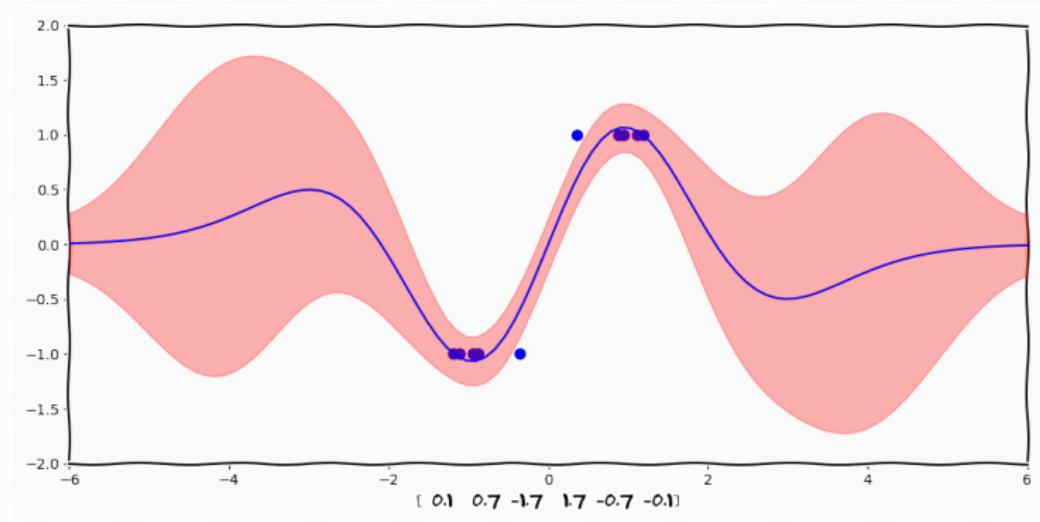
Composite Functions



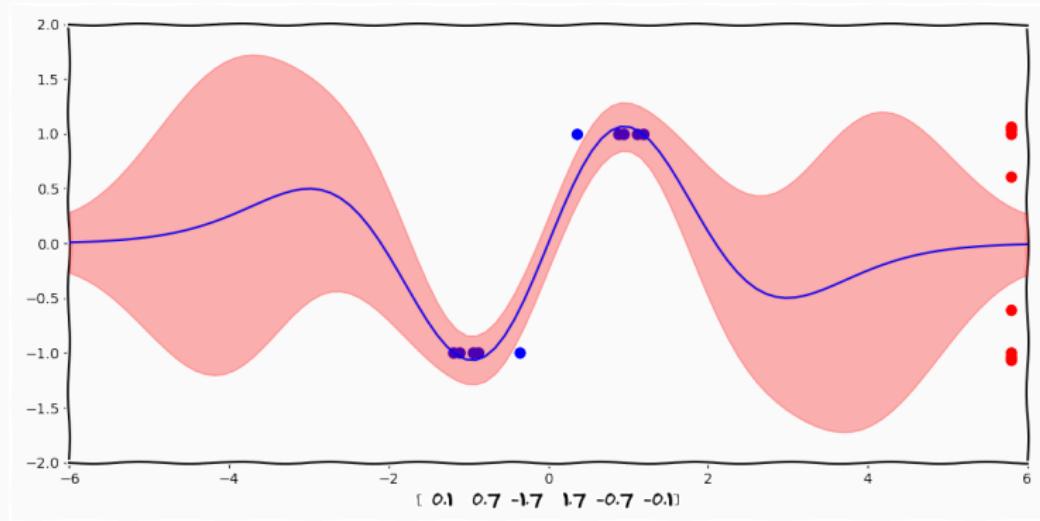
Composite Functions



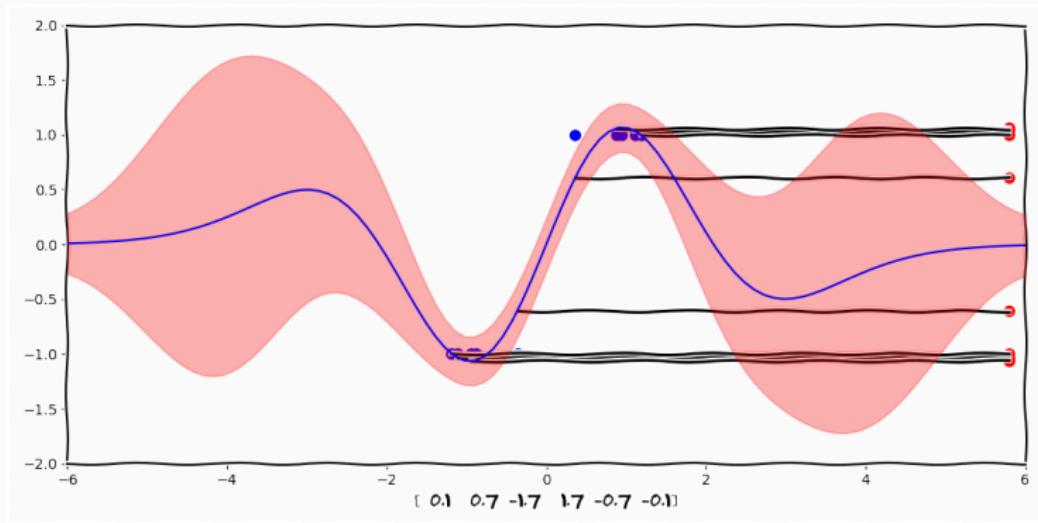
Composite Functions



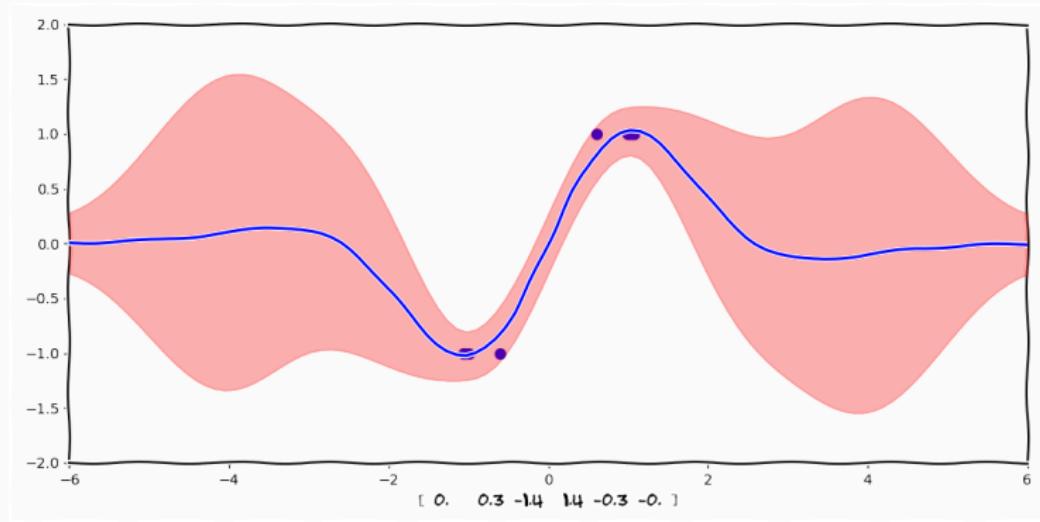
Composite Functions



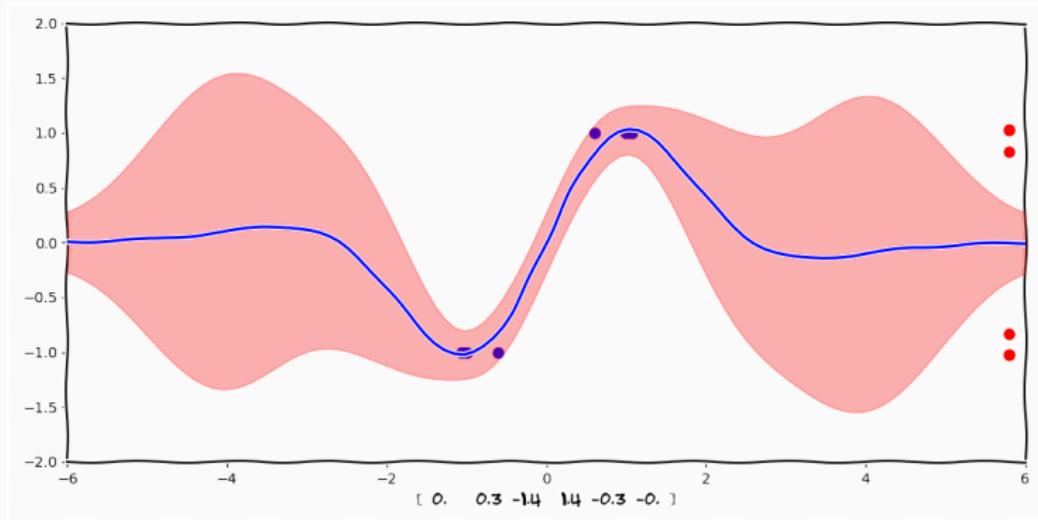
Composite Functions



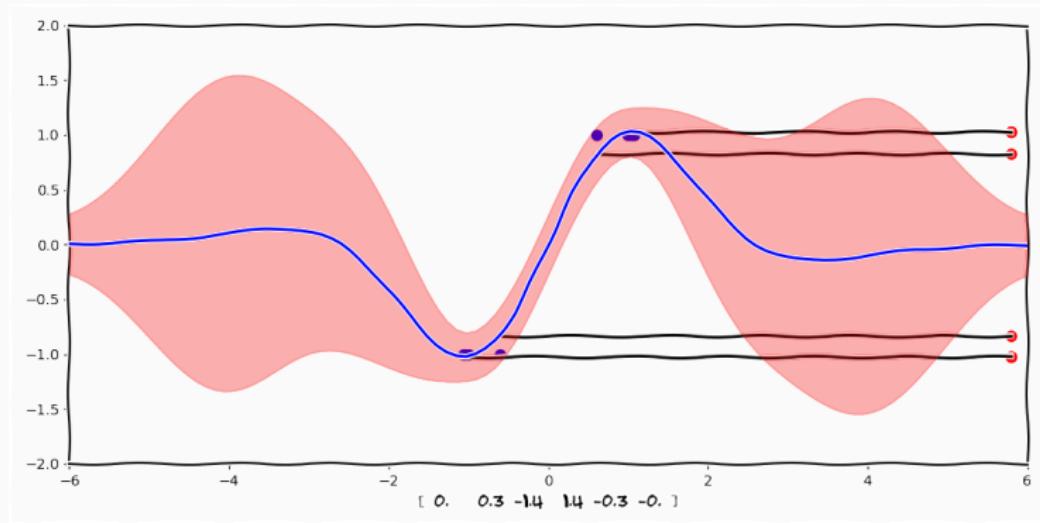
Composite Functions



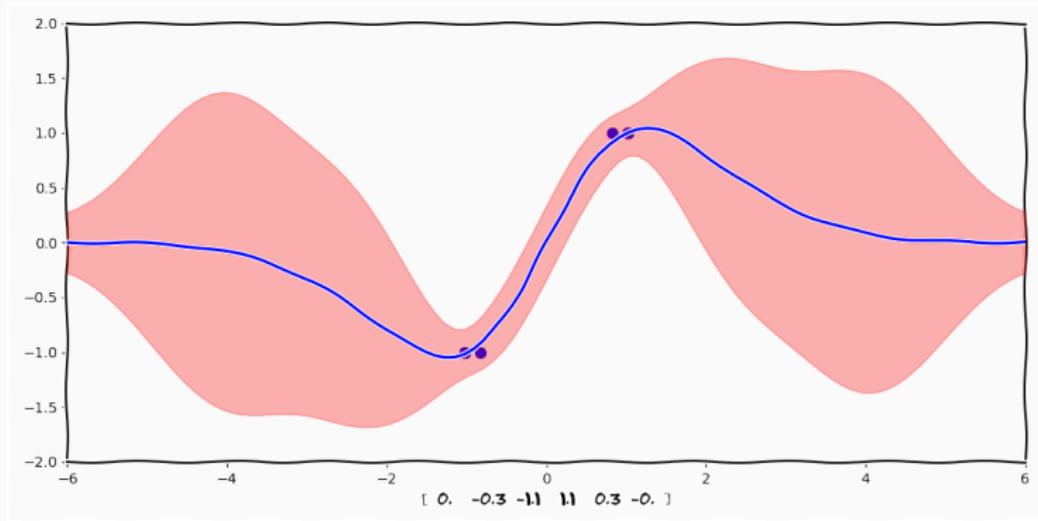
Composite Functions



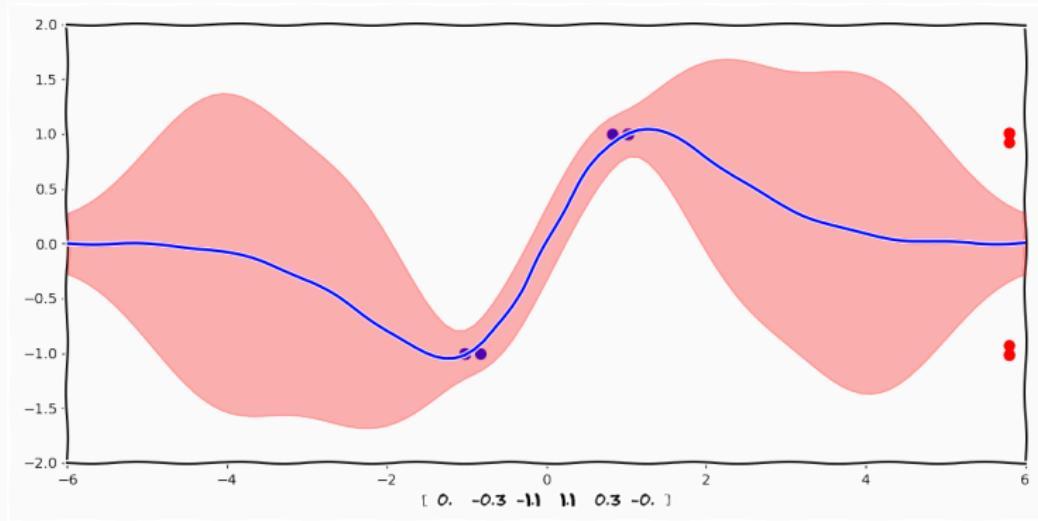
Composite Functions



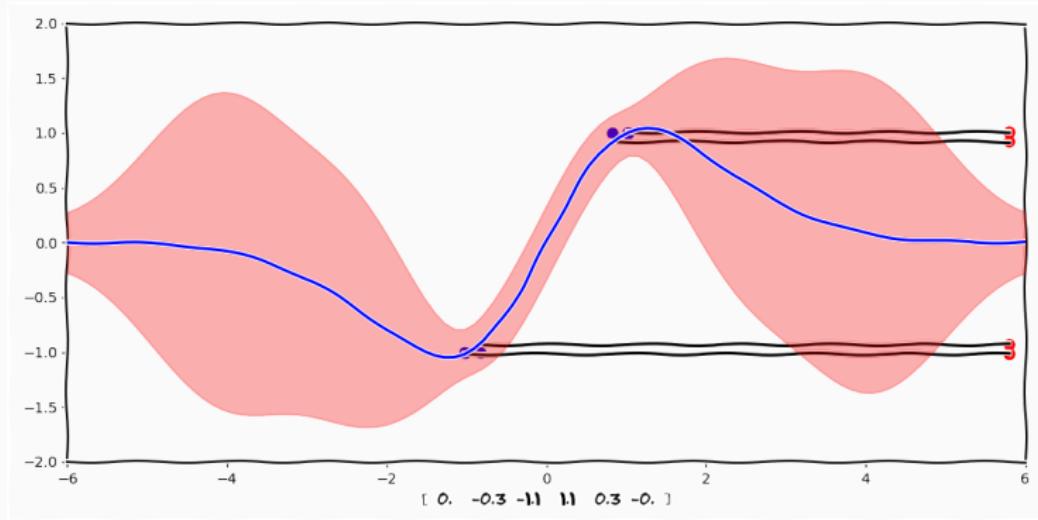
Composite Functions



Composite Functions



Composite Functions



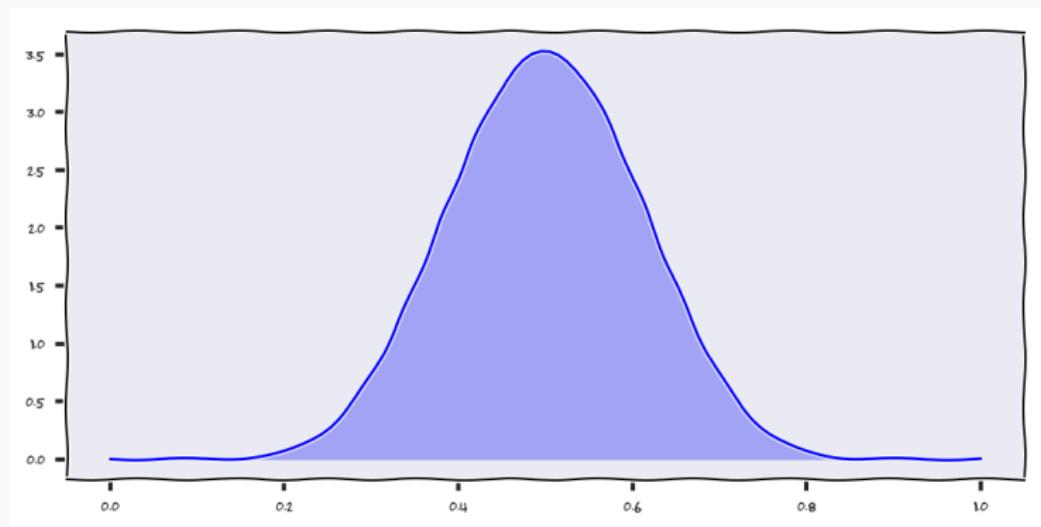
Theorem (Change of Variable)

Let $x \in \mathcal{X} \subseteq \mathbb{R}^n$ be a random vector with a probability density function given by $p_x(x)$, and let $y \in \mathcal{Y} \subseteq \mathbb{R}^n$ be a random vector such that $\psi(y) = x$, where the function $\psi : \mathcal{Y} \rightarrow \mathcal{X}$ is bijective of class of \mathcal{C}^1 and $|\bigtriangledown \psi(y)| > 0, \forall y \in \mathcal{Y}$. Then, the probability density function $p_y(\cdot)$ induced in \mathcal{Y} is given by

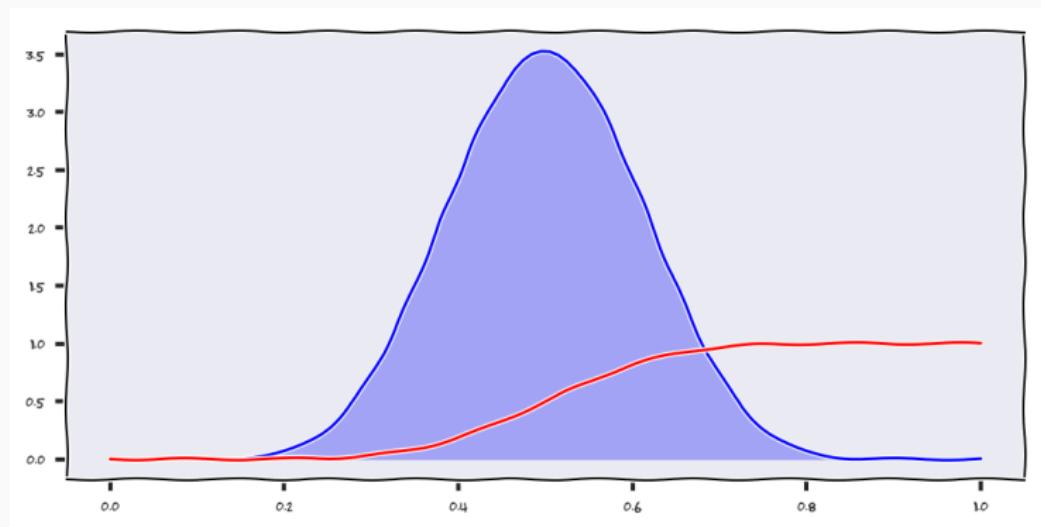
$$p_y(y) = p_x(\psi(y)) |\bigtriangledown \psi(y)|$$

where $\bigtriangledown \psi(\cdot)$ denotes the Jacobian of $\psi(\cdot)$, and $|\cdot|$ denotes the determinant operator.

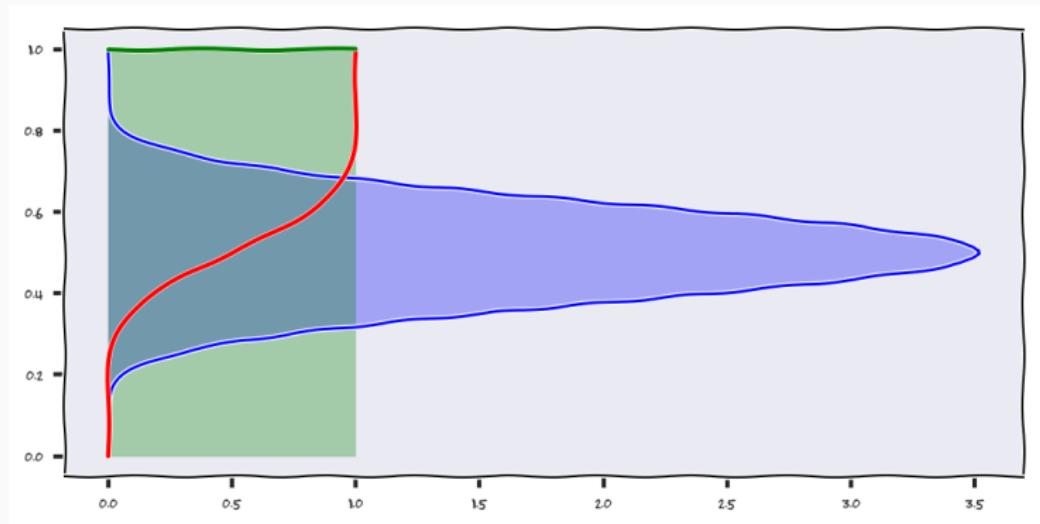
Sampling



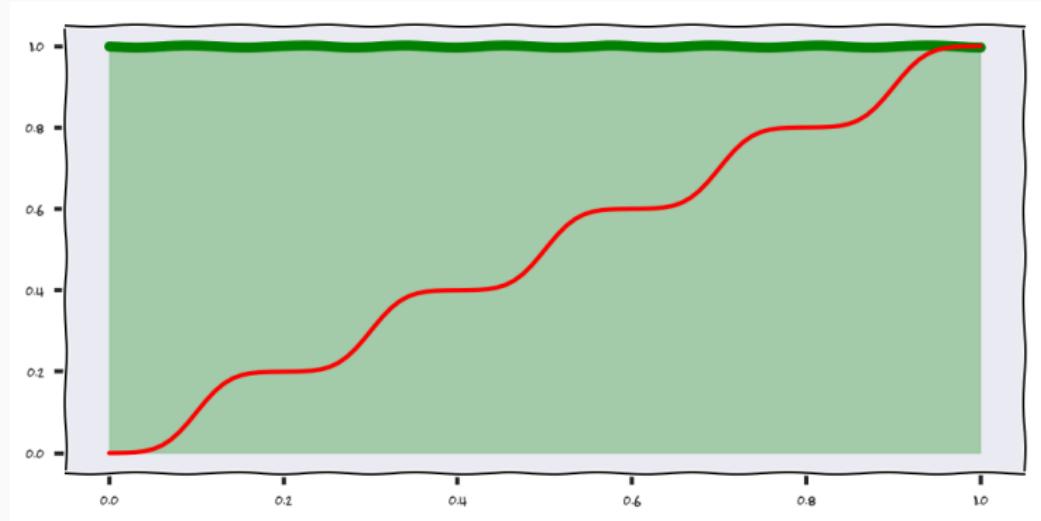
Sampling



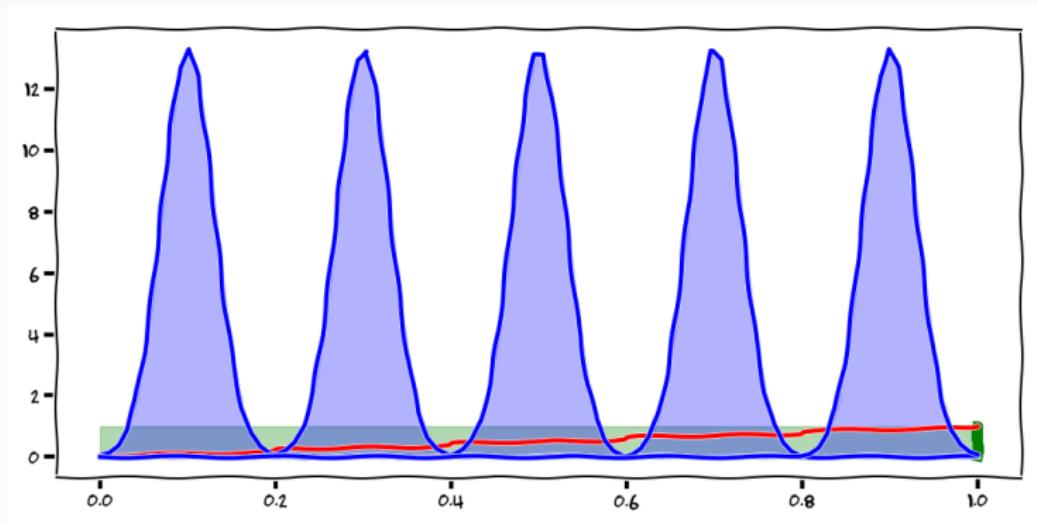
Sampling



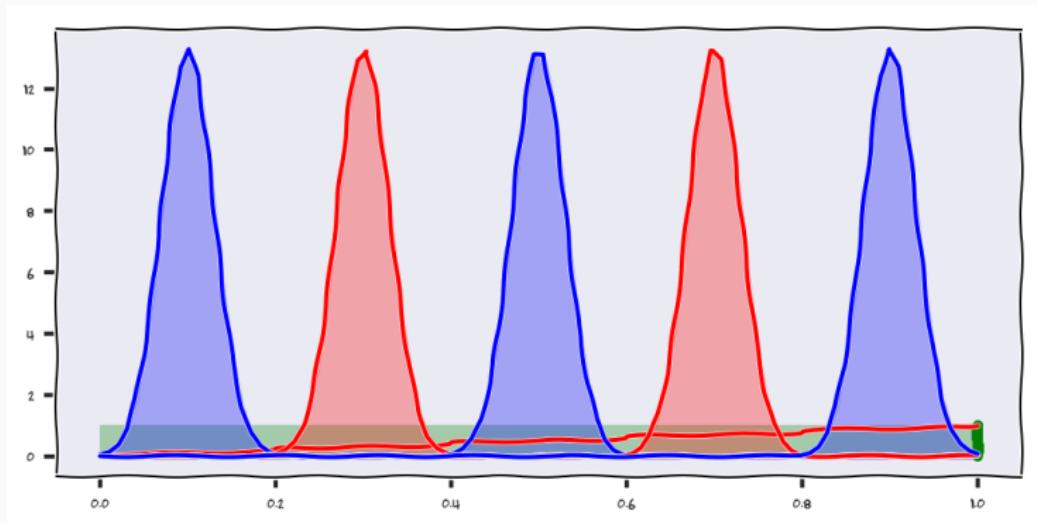
Change of Variables



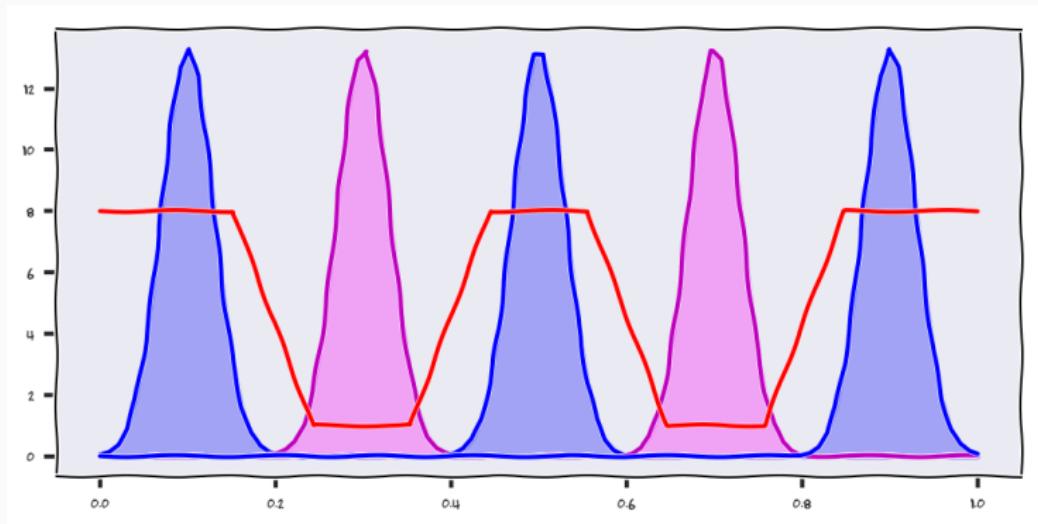
Change of Variables



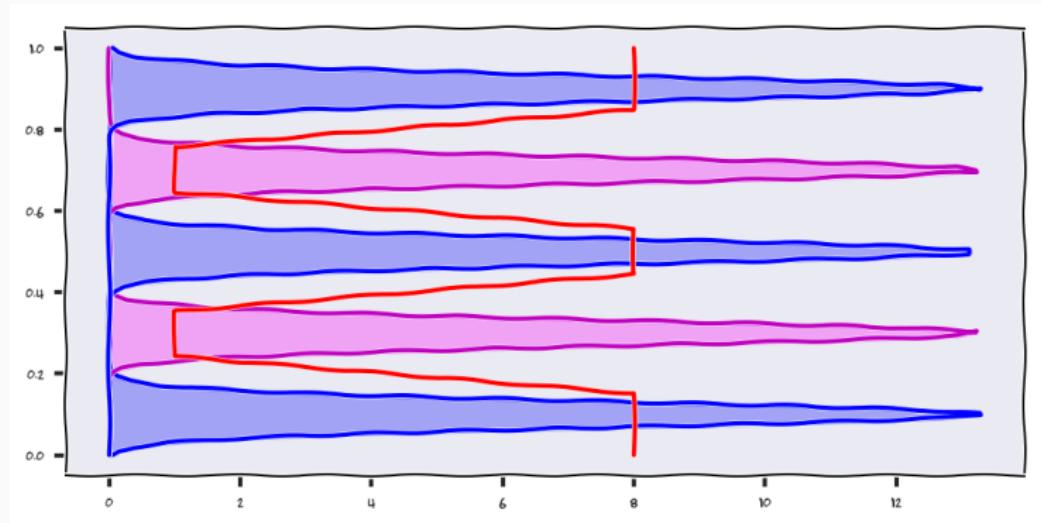
Change of Variables



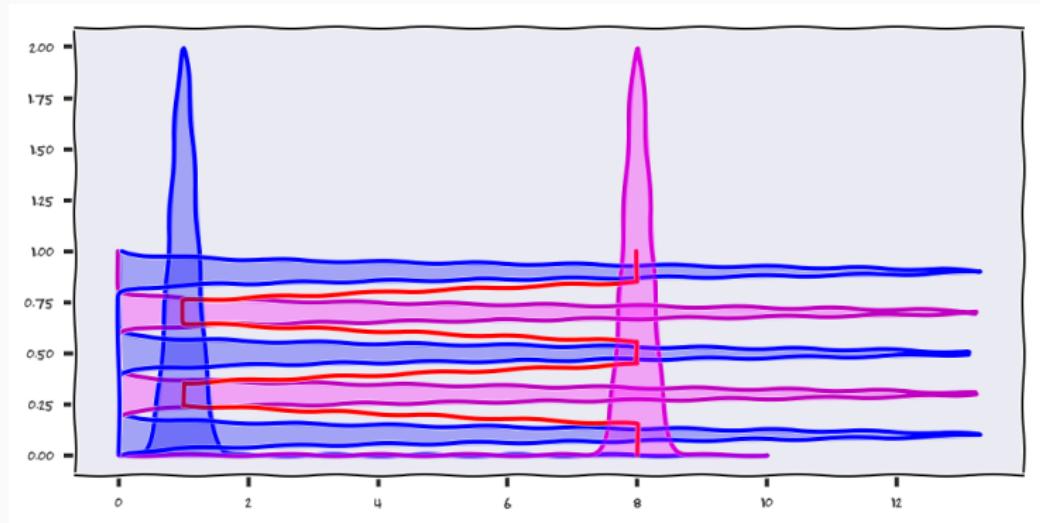
Change of Variables



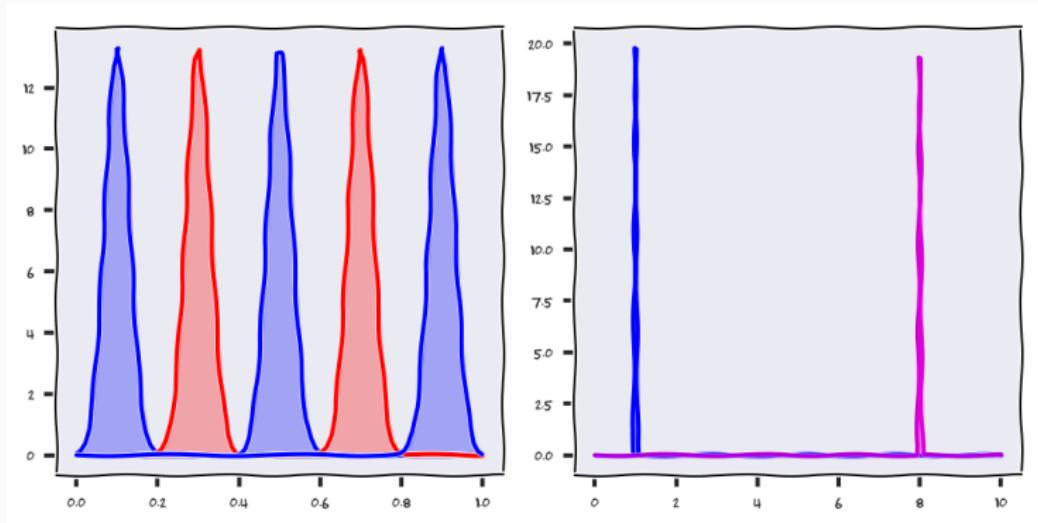
Change of Variables



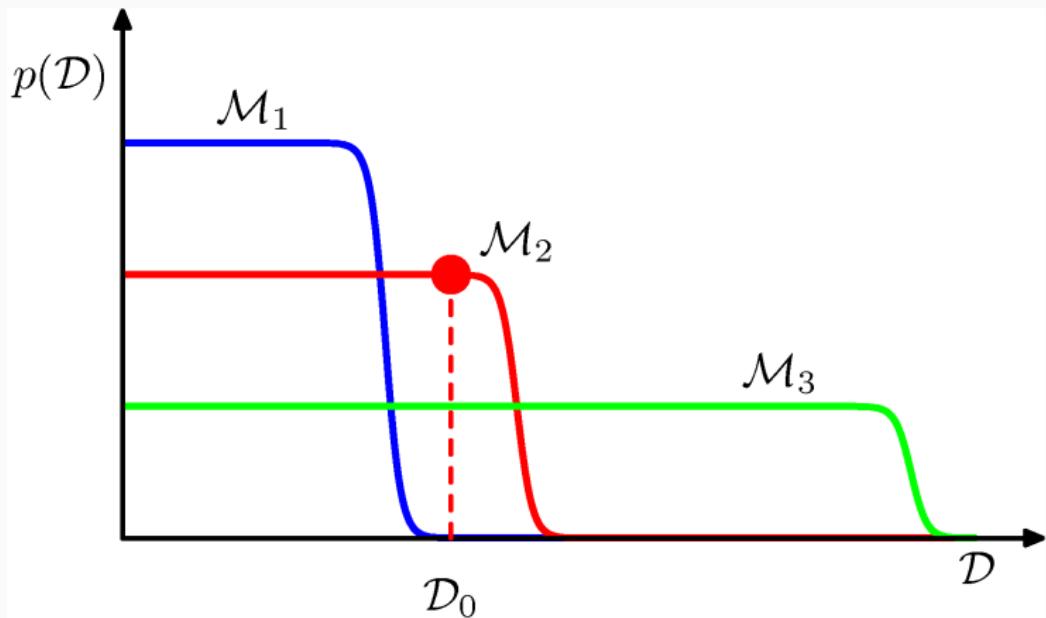
Change of Variables



Change of Variables



MacKay plot



Composite Functions

- With each consecutive composition we stretch and compress the space
- If the function is non-monotonic we collapse the space
- Each composition **reduces** the representative power of the final mapping

Learning

- Specifies a composite function

$$y_i = g(\mathbf{x}_i) = (f_K \circ f_{K-1} \circ \dots \circ f_1)(\mathbf{x}_i)$$

- Traditional form

$$y_k = g(\mathbf{x}, \mathbf{W}) = \sigma \left(\sum_j^M w_{kj}^{(2)} h \left(\sum_j^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

- Linear weights, non-linear activation functions (sigmoid, tanh, etc)

- Forward propagation

$$g : \mathcal{X} \rightarrow \mathcal{Y}$$

- Forward propagation results in error

$$E(\mathbf{W}) = \sum_n^N \frac{1}{2} (g(\mathbf{x}_n, \mathbf{W}) - y_n)^2 + \text{reg}(\mathbf{W})$$

- Propagate error back to update \mathbf{W}

Backpropagation

- "Invented" many times but usually attributed to Rummelhart 1986
- Gradient based update of weights

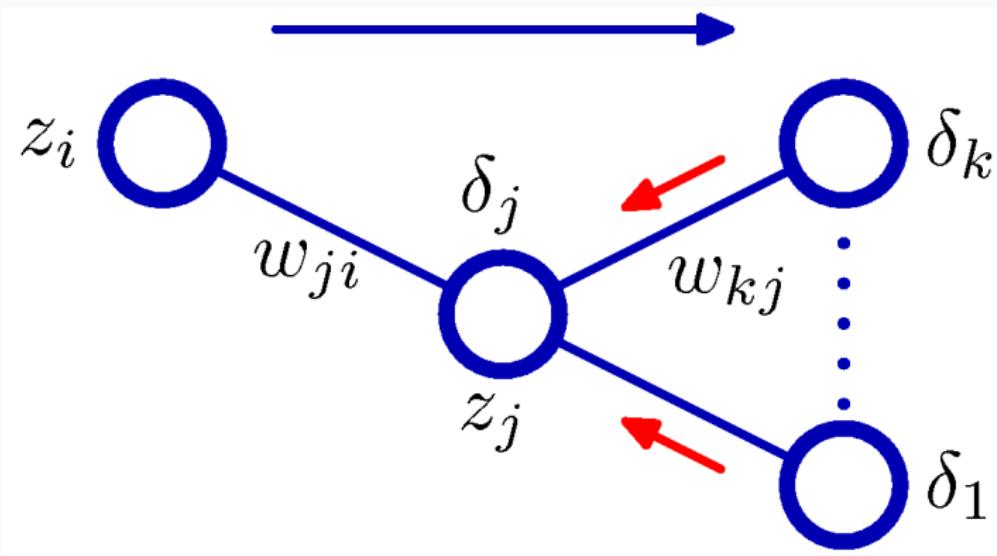
$$\frac{\partial}{\partial \mathbf{W}} E$$

- Iteratively updated weights

$$\mathbf{W}^{(t)} = \mathbf{W}^{(t-1)} + \eta \frac{\partial}{\partial \mathbf{W}} E,$$

where η is the learning rate

Backpropagation



$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$$

Backpropagation

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} = \frac{\partial h(a_j)}{\partial a_j} \sum_k w_{kj} \delta_k$$

- We know the error on the output layer
- The formula above allows us to **propagate** this error back through the layers

Backpropagation

1. Randomly initialise weights
2. Forward pass (get output error)
3. Compute gradients for one step back
4. Iteratively push gradients back
5. Update each layer
6. Run till convergence (or well. . .)

Activation functions

- Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

- Tanh

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

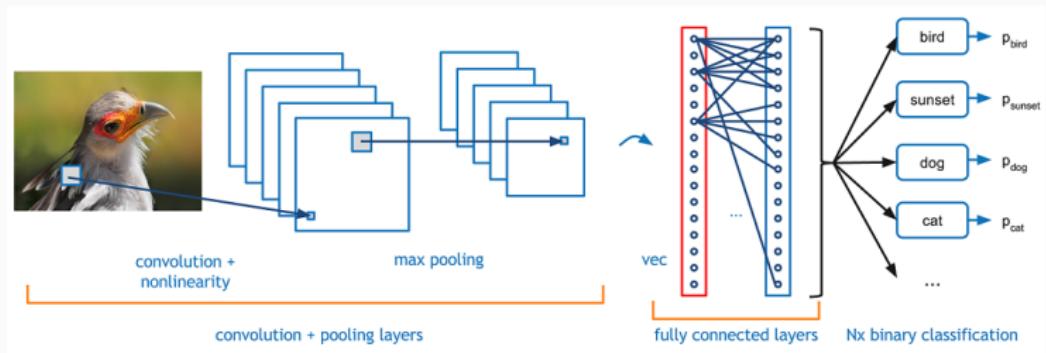
- ReLu

$$f(x) = \max(0, x) \approx \log(1 + e^x)$$

Architectures

- Backpropagation allows for updates of any differentiable composite function
- With auto-diff this means that you can **hack** together lots of structures
- If we know a-priori what we want to learn we can reduce the search space
- We have to specify everything in terms of hard constraints

Convolutional Neural Networks



- Designed by Yann Lecun late eighties, did well on some tasks but really took off when data grew
- Includes two special types of transformations
 - Convolutions: linear transformation
 - Pooling: non-linear transformation
- Highly designed structures

Convolutional Neural Networks

- Convolution

$$s(x) = \int x(t)w(x-t)dt$$

- A convolution takes a weighted average within a spatial region (eg. blurring)

$$S_{ij} = \sum_m \sum_n I(i-m, j-n)K(m, n)$$

- Makes a lot of sense for images, average nearby pixels

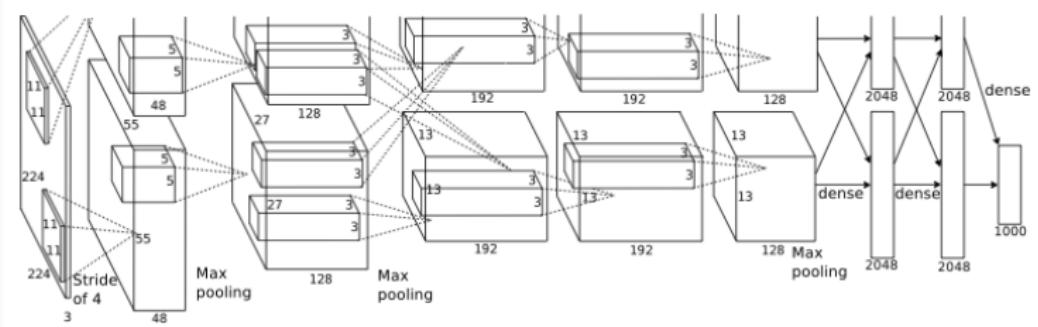
Convolutional Neural Networks

- Pooling

$$S_{ij} = \max_{i=1, j=1}^{i=N, j=M} \{x_{ij}\}$$

- Introduces invariances
 - translation
 - scaling
- *it is more important that something is present than where it is present*

Convolutional Neural Networks



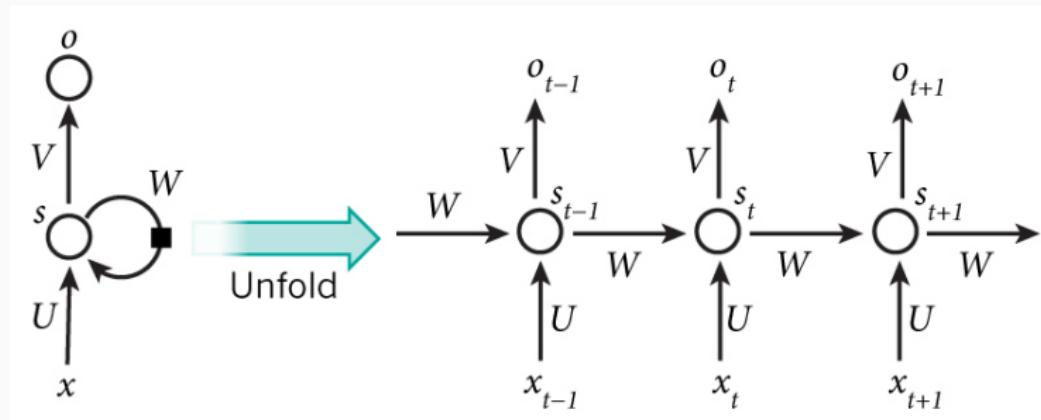
Recurrent Neural Networks

- Recurrent Neural Networks designed to model sequential data
- Dynamical system

$$s^{(t)} = f(s^{(t-1)}, x^{(t)})$$

- Idea:
 - unfold the computation into a single graph
 - becomes a standard neural network

Recurrent Neural Networks²



²[http://www.wildml.com/2015/09/](http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/)

[recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/](http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/)

Architectures

- There exists many many different architectures

³Chomsky, N. A., & Fodor, J. A. (1980). The inductivist fallacy.

Architectures

- There exists many many different architectures
- Few are derived or justified on any scientific basis

³Chomsky, N. A., & Fodor, J. A. (1980). The inductivist fallacy.

Architectures

- There exists many many different architectures
- Few are derived or justified on any scientific basis
- There are many post-justifications³

³Chomsky, N. A., & Fodor, J. A. (1980). The inductivist fallacy.

- There exists many many different architectures
- Few are derived or justified on any scientific basis
- There are many post-justifications³
- They do work really well but don't listen to the pseudo-science
dare to have your own intuitions instead

³Chomsky, N. A., & Fodor, J. A. (1980). The inductivist fallacy.

The Problem

Decision vs. Understanding

- Neural networks, they way they are described are discriminative this means

Decision vs. Understanding

- Neural networks, they way they are described are discriminative this means
 - they do not model the data

Decision vs. Understanding

- Neural networks, they way they are described are discriminative this means
 - they do not model the data
 - they do not understand the data

Decision vs. Understanding

- Neural networks, they way they are described are discriminative this means
 - they do not model the data
 - they do not understand the data
 - there is no such thing as uncertainty

Decision vs. Understanding

- Neural networks, they way they are described are discriminative this means
 - they do not model the data
 - they do not understand the data
 - there is no such thing as uncertainty
- They are not models they are decision machines

Decision vs. Understanding

- Neural networks, they way they are described are discriminative this means
 - they do not model the data
 - they do not understand the data
 - there is no such thing as uncertainty
- They are not models they are decision machines
- *Like only revising old exams*



track cycling
cycling
track cycling
road bicycle racing
marathon
ultramarathon



ultramarathon
ultramarathon
half marathon
running
marathon
inline speed skating



heptathlon
heptathlon
decathlon
bundles
pentathlon
sprint (running)



bikejoring
mushing
bikejoring
harness racing
skijoring
carting



longboarding
longboarding
aggressive inline skating
freestyle scootering
freestyle (skateboard)
sandboarding



ultimate (sport)
ultimate (sport)
hurling
flag football
association football
rugby sevens



demolition derby
demolition derby
monster truck
mud bogging
motocross
grand prix motorcycle racing



telemark skiing
snowboarding
telemark skiing
nordic skiing
ski touring
skijoring



whitewater kayaking
rafting
kayaking
canoeing
adventure racing



arena football
indoor american football
arena football
canadian football
american football
women's lacrosse

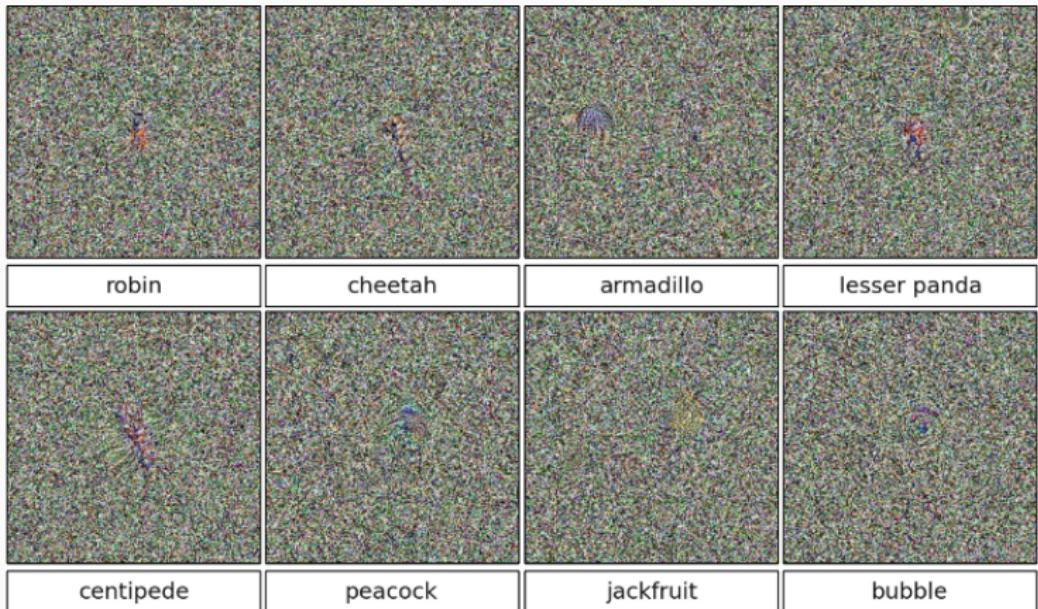


reining
barrel racing
rodeo
reining
cowboy action shooting
ball riding



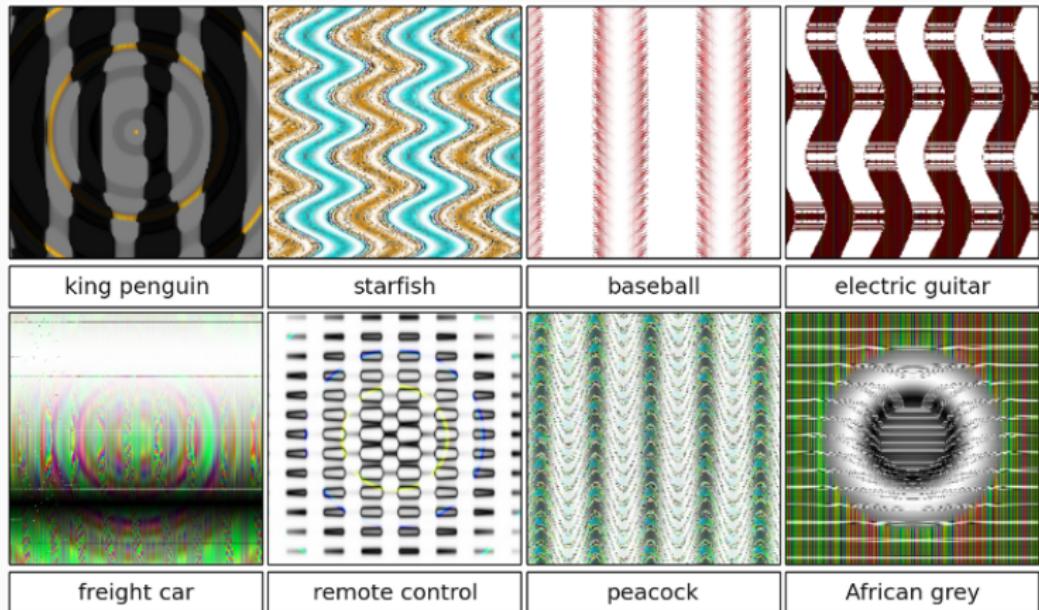
eight-ball
nine-ball
blackball (pool)
trick shot
eight-ball
straight pool

Fools⁴



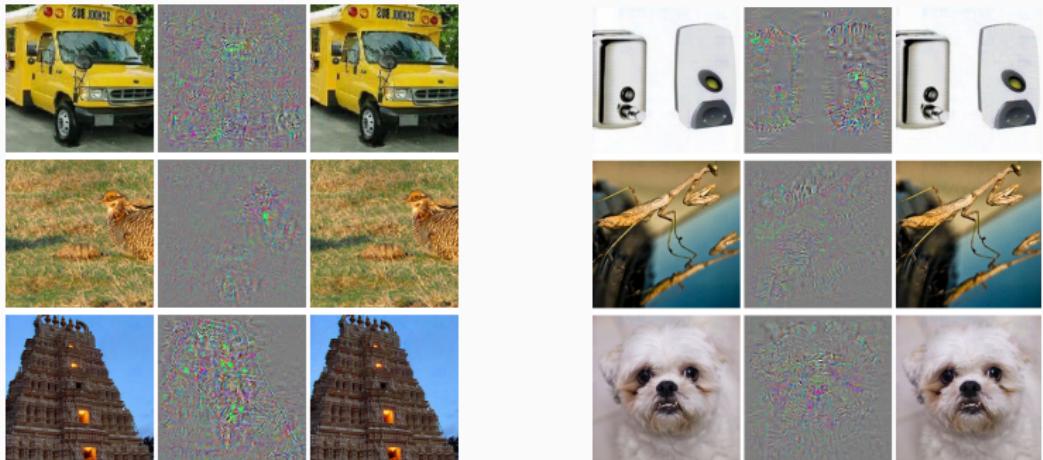
⁴http://www.evolvingai.org/files/DNNsEasilyFooled_cvpr15.pdf

Fools⁴



⁴http://www.evolvingai.org/files/DNNsEasilyFooled_cvpr15.pdf

Fools⁴



Left correctly classified, all right images are classified as "ostrich"

⁴http://www.evolvingai.org/files/DNNsEasilyFooled_cvpr15.pdf

Fools⁵



SHIP
CAR(99.7%)



HORSE
FROG(99.9%)



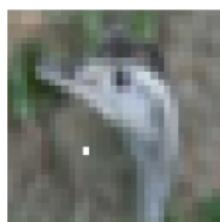
DEER
AIRPLANE(85.3%)



HORSE
DOG(70.7%)



DOG
CAT(75.5%)



BIRD
FROG(86.5%)

⁵Su, J., Vargas, D. V., & Kouichi, S., One pixel attack for fooling deep neural networks,(2017).

The Black-arts

- Early stopping

The Black-arts

- Early stopping
- Layer-wise training

The Black-arts

- Early stopping
- Layer-wise training
- Denoising

The Black-arts

- Early stopping
- Layer-wise training
- Denoising
- Adveserial training

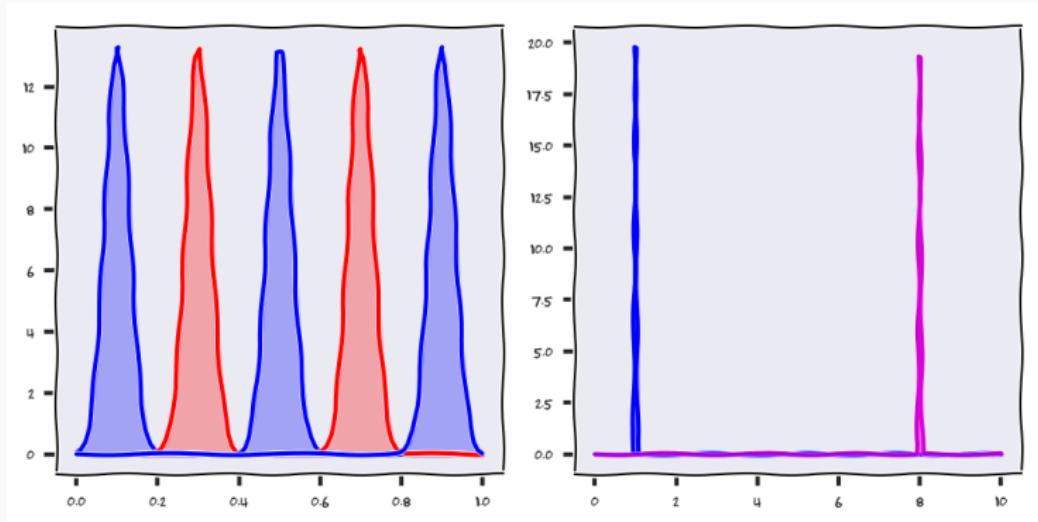
The Black-arts

- Early stopping
- Layer-wise training
- Denoising
- Adveserial training
- Dropout

The Black-arts

- Early stopping
- Layer-wise training
- Denoising
- Adveserial training
- Dropout
- Etc.

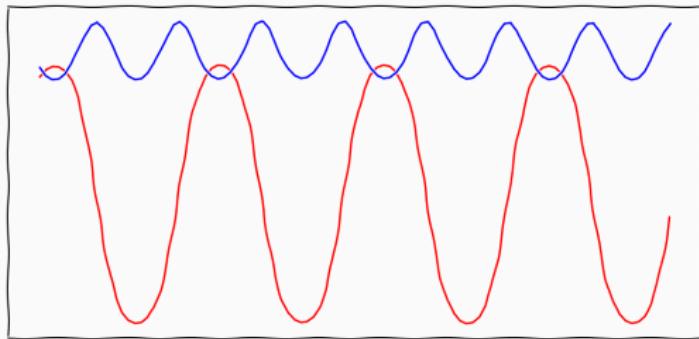
Change of Variables



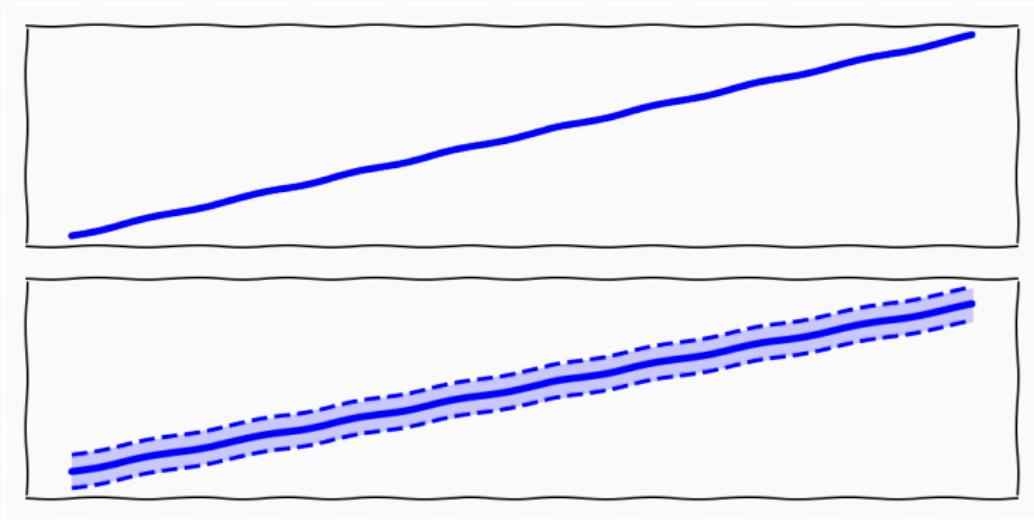
Decision machines

- Decision machines are useful
 - logistic regression, Support Vector Machines
 - nearest neighbours, etc.
- These methods do not warp the input space to the same extent
 - do not mess with proximity to the same extent
- You need enormous amounts of data to make sure you retain representation of space

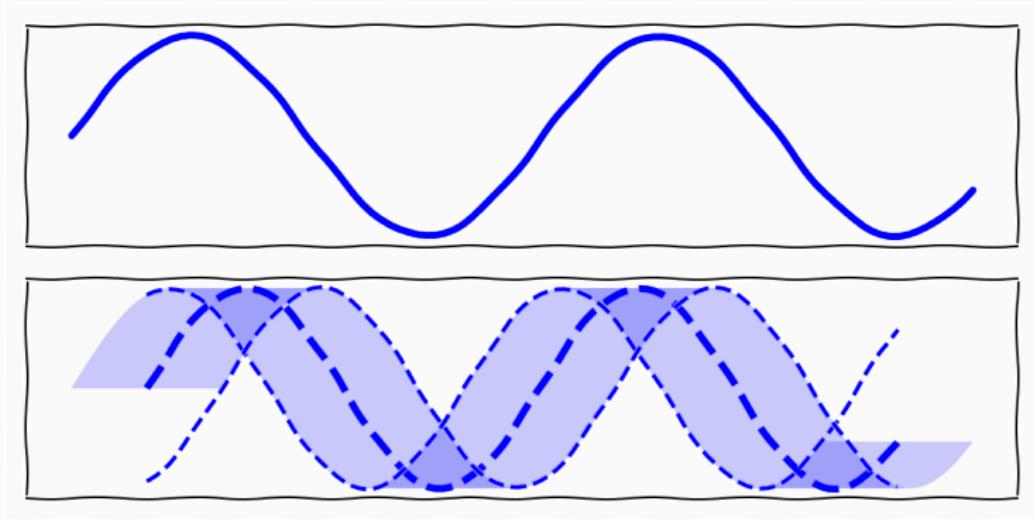
Composition: priors



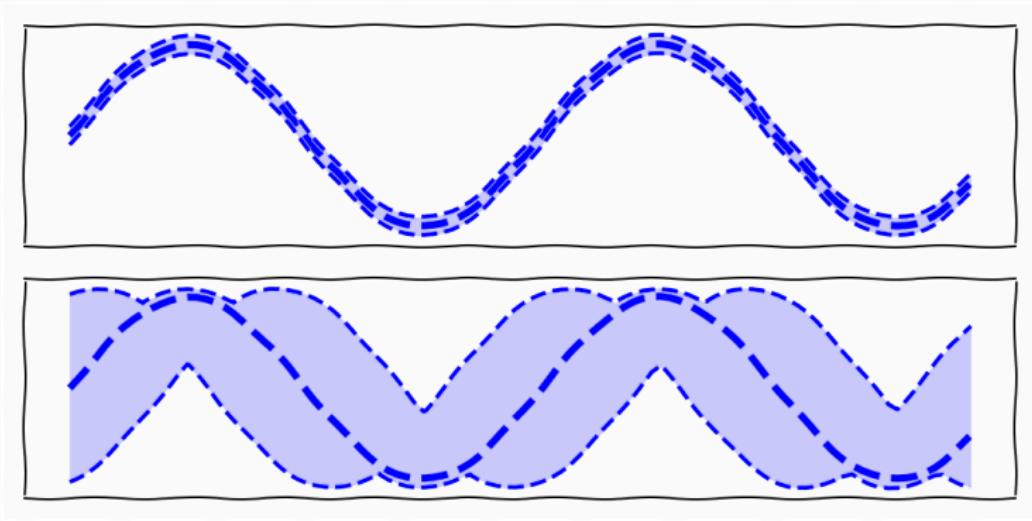
Composition: uncertainty



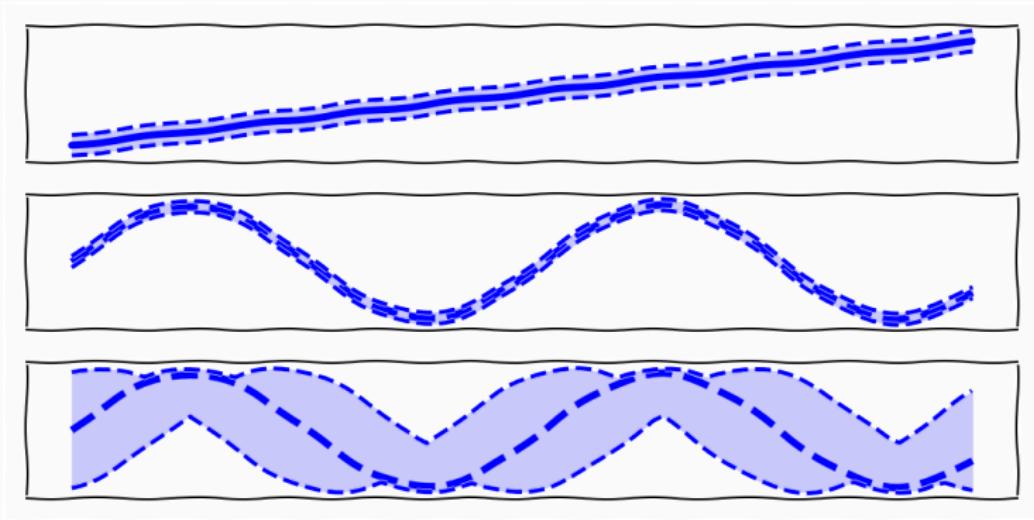
Composition: uncertainty



Composition: uncertainty



Composition: uncertainty



Summary

Summary

- Neural networks is nothing special, forget the neuroscience, it has nothing to do with it

Summary

- Neural networks is nothing special, forget the neuroscience, it has nothing to do with it
- Composite functions **cannot** model more things

Summary

- Neural networks is nothing special, forget the neuroscience, it has nothing to do with it
- Composite functions **cannot** model more things
- However,

Summary

- Neural networks is nothing special, forget the neuroscience, it has nothing to do with it
- Composite functions **cannot** model more things
- However,
 - they can easily warp the input space to model **less** things

Summary

- Neural networks is nothing special, forget the neuroscience, it has nothing to do with it
- Composite functions **cannot** model more things
- However,
 - they can easily warp the input space to model **less** things
 - they are very simple and parallel

Summary

- Neural networks is nothing special, forget the neuroscience, it has nothing to do with it
- Composite functions **cannot** model more things
- However,
 - they can easily warp the input space to model **less** things
 - they are very simple and parallel
 - computationally trivial

Summary

- Neural networks is nothing special, forget the neuroscience, it has nothing to do with it
- Composite functions **cannot** model more things
- However,
 - they can easily warp the input space to model **less** things
 - they are very simple and parallel
 - computationally trivial
- However,

Summary

- Neural networks is nothing special, forget the neuroscience, it has nothing to do with it
- Composite functions **cannot** model more things
- However,
 - they can easily warp the input space to model **less** things
 - they are very simple and parallel
 - computationally trivial
- However,
 - lacks principle so hard to know what they do or how to do new things

Summary

- Neural networks is nothing special, forget the neuroscience, it has nothing to do with it
- Composite functions **cannot** model more things
- However,
 - they can easily warp the input space to model **less** things
 - they are very simple and parallel
 - computationally trivial
- However,
 - lacks principle so hard to know what they do or how to do new things
- With sufficient data we do not need to "learn" look-up is sufficient

eof

References

 David M Blei, Andrew Y Ng, and Michael I Jordan.
Latent dirichlet allocation.

3:993–1022, March 2003.