Machine Learning

Neural Networks

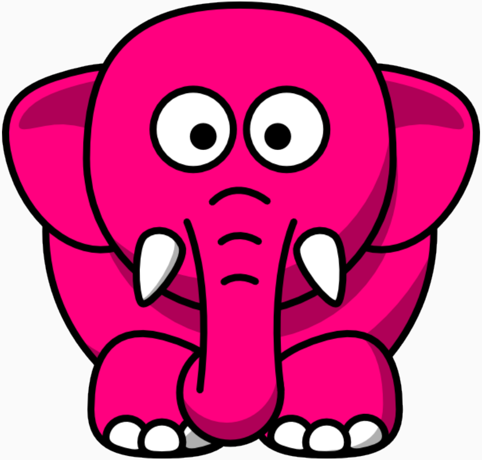
Carl Henrik Ek - carlhenrik.ek@bristol.ac.uk

November 26, 2018

http://www.carlhenrik.com

Introduction

Today



Darthmouth workshop[1]

[1] https://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

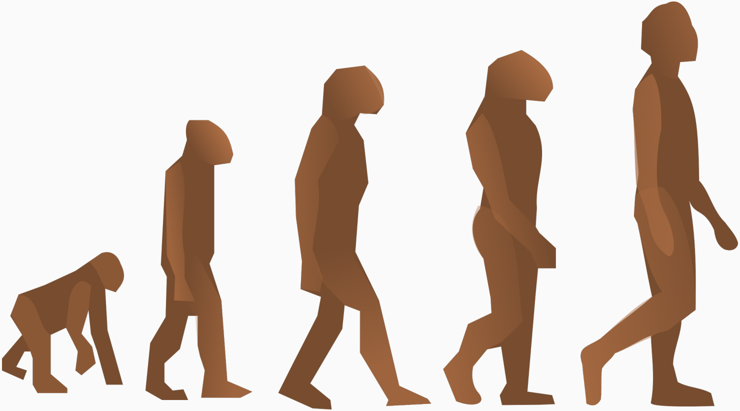
AI

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

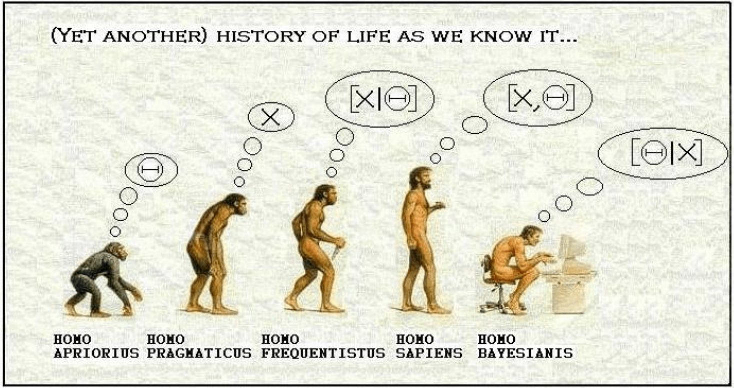
AI

- 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



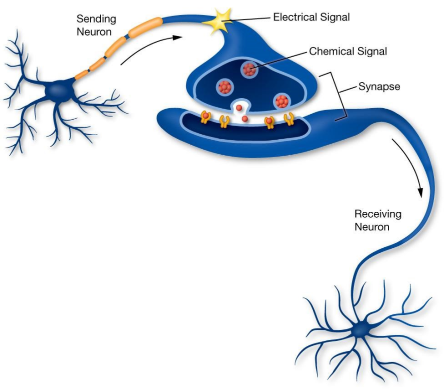
Another view



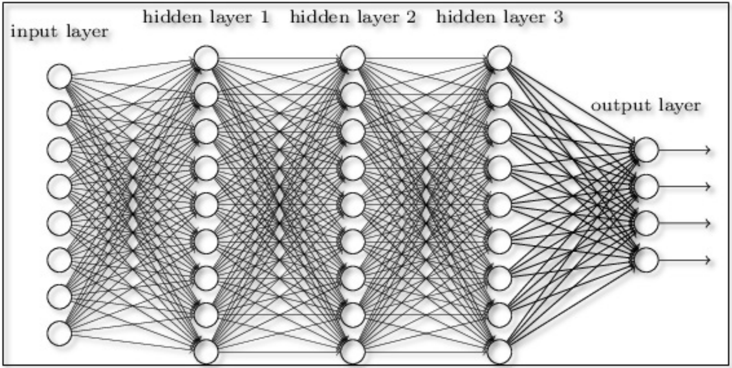
Neural Networks

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

Neurons and spike trains

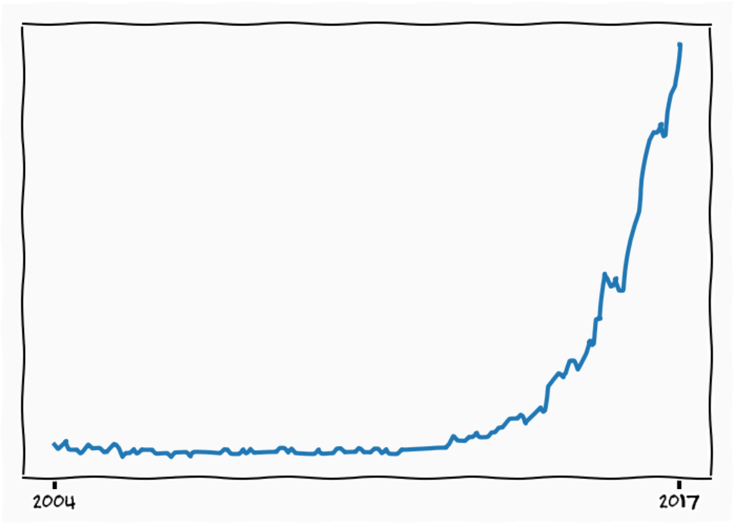


Neural Networks



AI Winter

Media



Neural Networks

Functions

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

- regression: f is identitiy 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

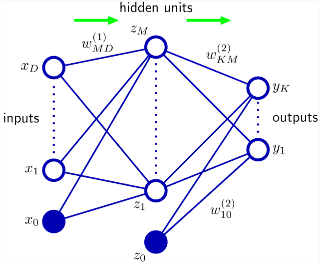
$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

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

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?

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

Linear Algebra

- Kernel of a function

$$\text{Kern}(f_k) = \left\{(\mathsf{x}, \mathsf{x}')|f_k(\mathsf{x}) = f_k(\mathsf{x}')\right\}$$

- Kernel of a function

$$\mathrm{Kern}(f_k) = \left\{(\mathbf{x}, \mathbf{x}') | f_k(\mathbf{x}) = f_k(\mathbf{x}')\right\}$$

- Image of a function

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

- Rank-Nullity Theorem

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

Linear Algebra

- Rank-Nullity Theorem

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

- Kernel of function

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

Linear Algebra

- Rank-Nullity Theorem

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

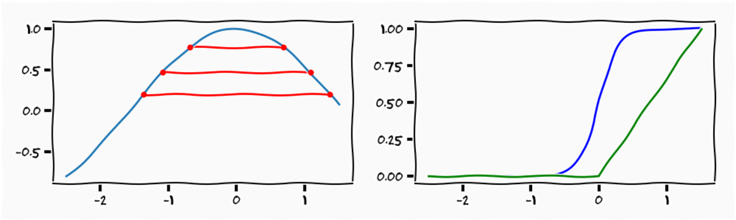
- Kernel of function

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

- Image of a function

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

Composition functions

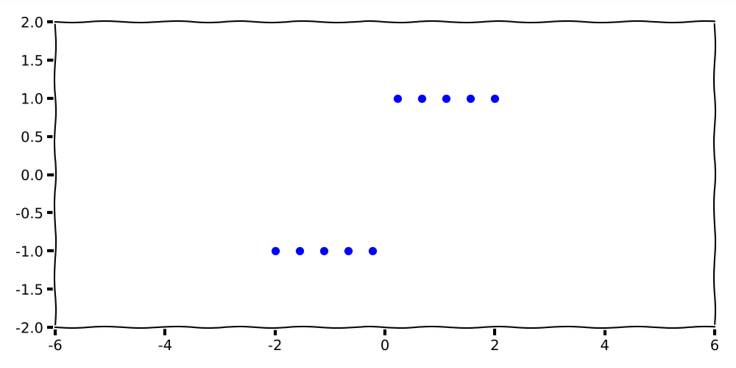


$$y = f_k(f_{k-1}(\ldots f_0(x))) = f_k \circ f_{k-1} \circ \cdots \circ f_1(x)$$

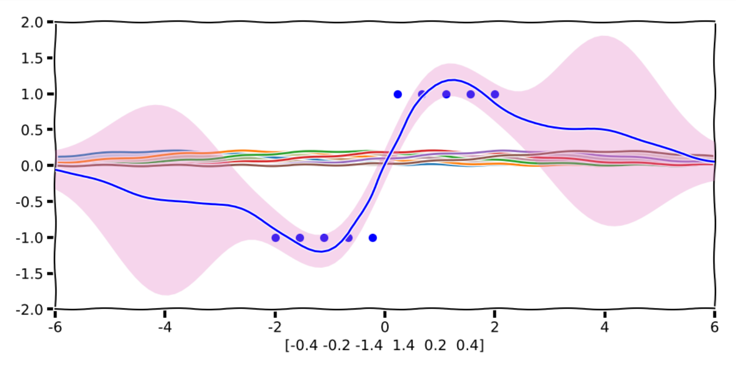
$$\mathrm{Kern}(f_1) \subseteq \mathrm{Kern}(f_{k-1} \circ \ldots \circ f_2 \circ f_1) \subseteq \mathrm{Kern}(f_k \circ f_{k-1} \circ \ldots \circ f_2 \circ f_1)$$

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

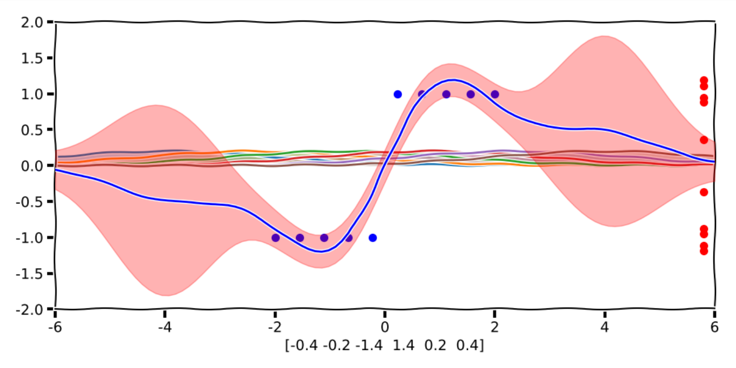
Composite Functions



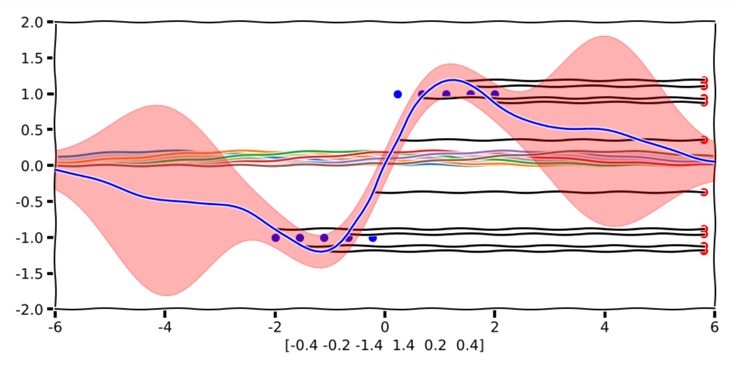
Composite Functions



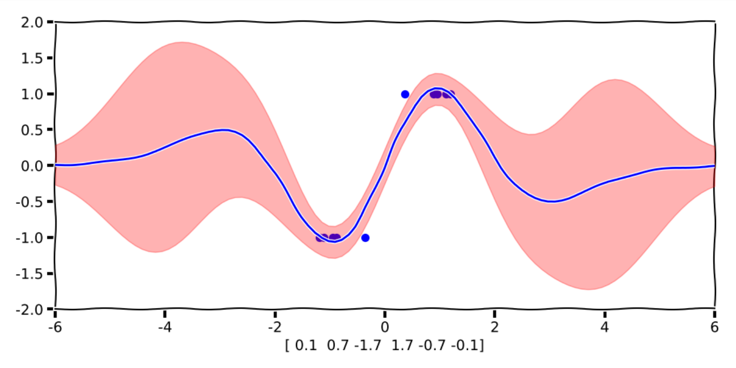
Composite Functions



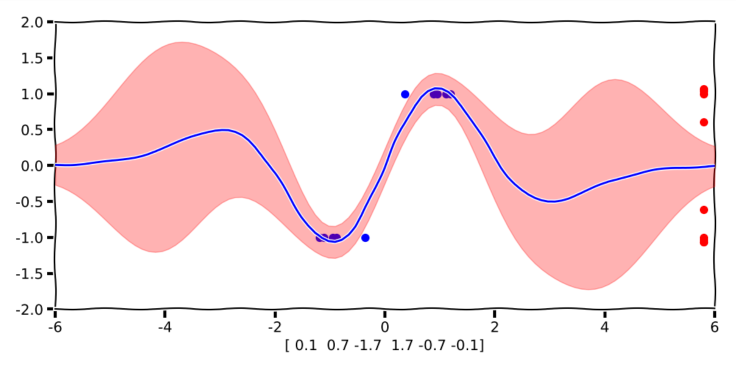
Composite Functions



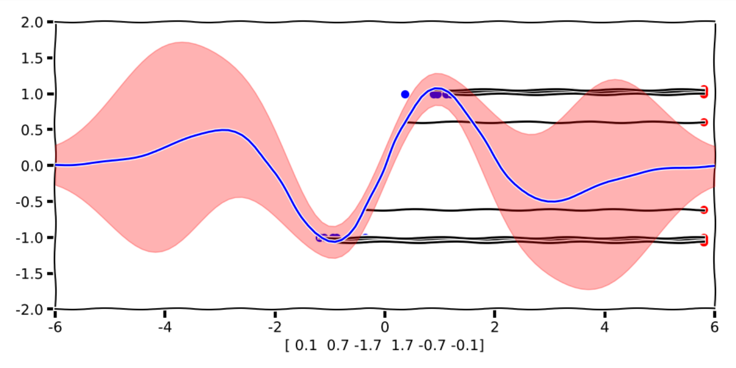
Composite Functions



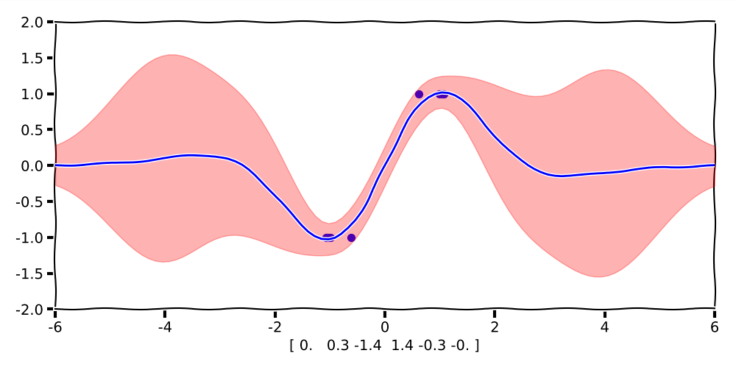
Composite Functions



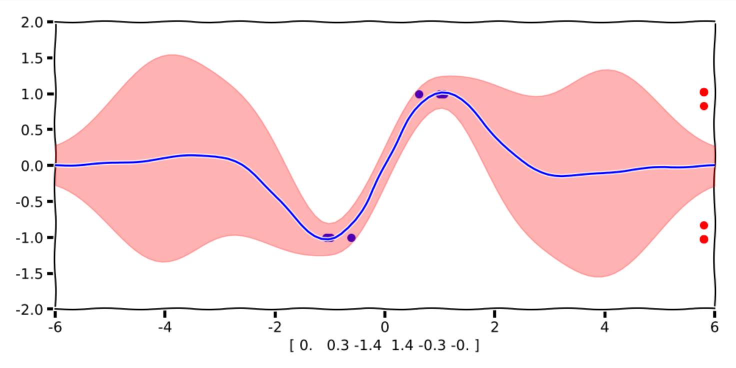
Composite Functions



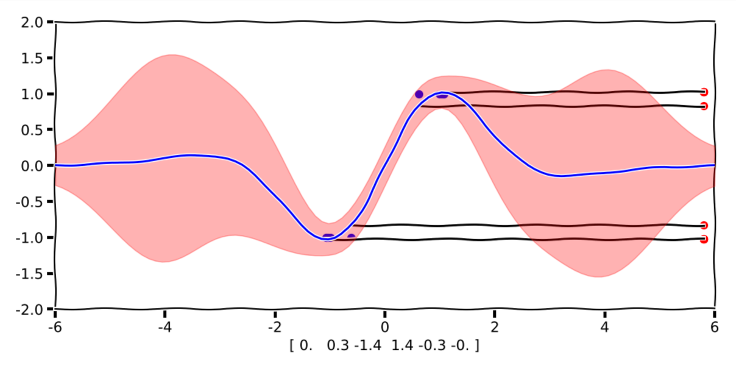
Composite Functions



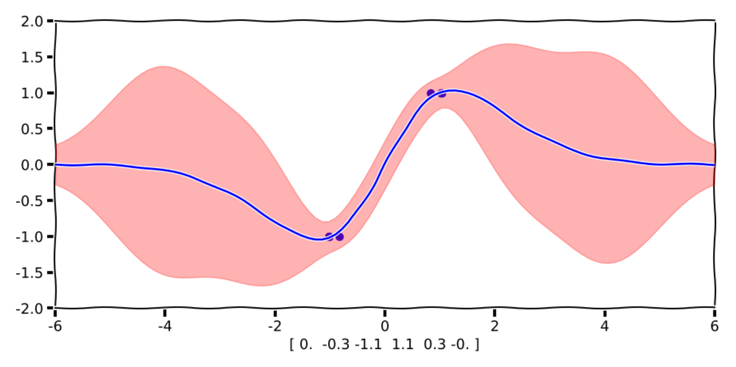
Composite Functions



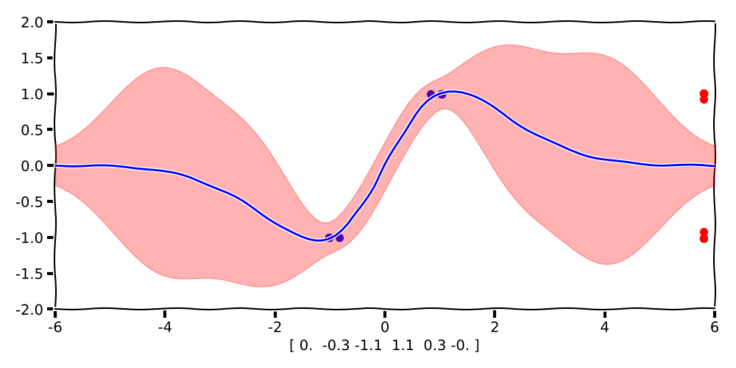
Composite Functions



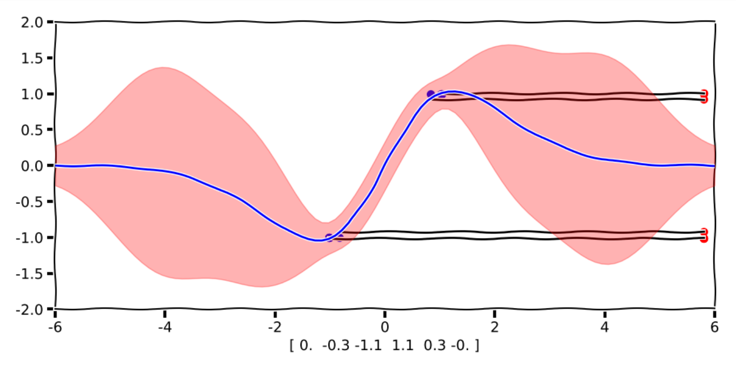
Composite Functions



Composite Functions



Composite Functions



Interpretability

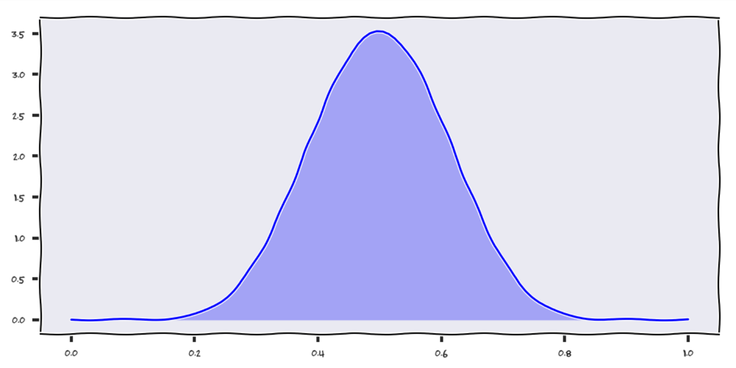
Theorem (Change of Variable)

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

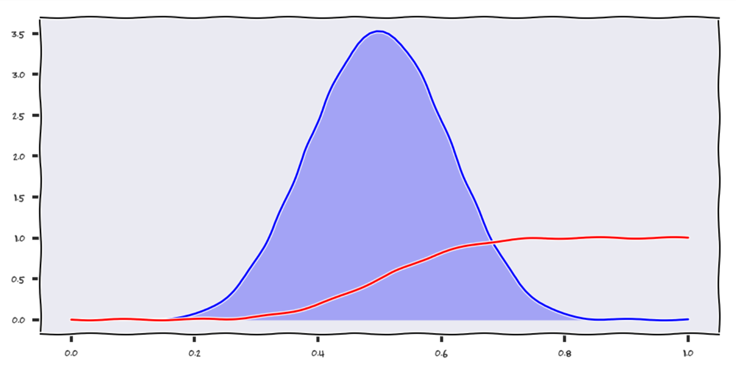
$$p_y(y) = p_x(\psi(y))| \bigtriangledown \psi(y)|$$

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

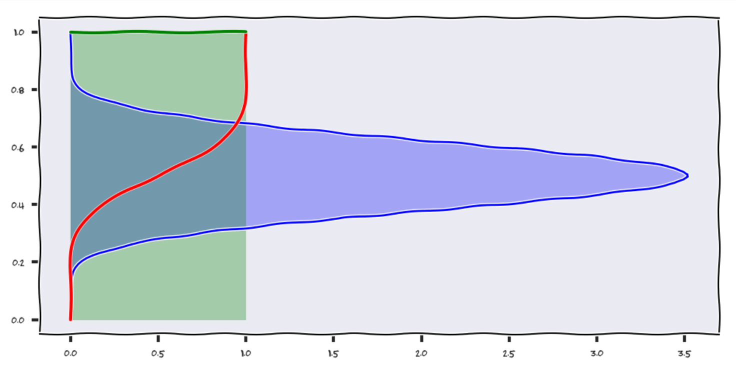
Sampling



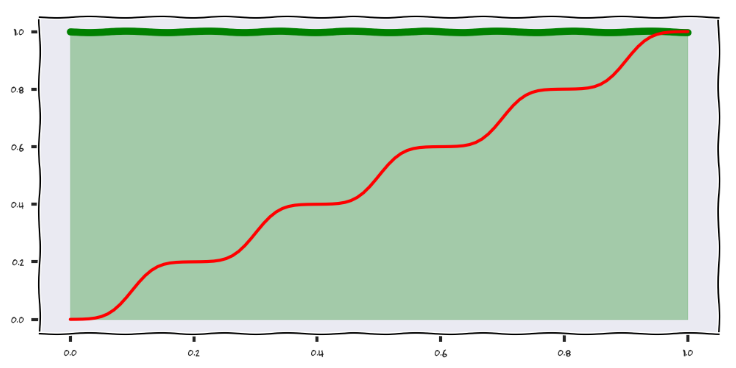
Sampling



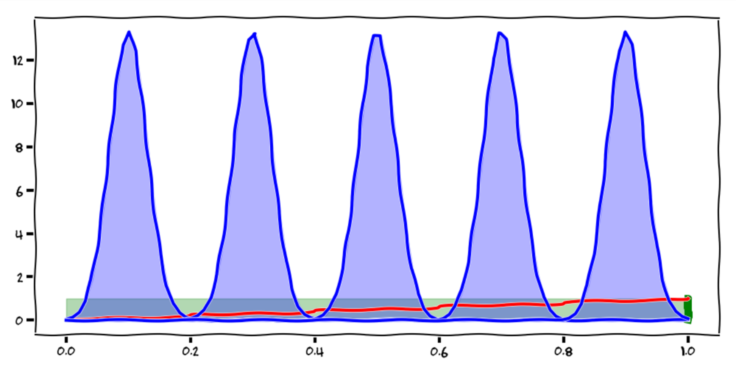
Sampling



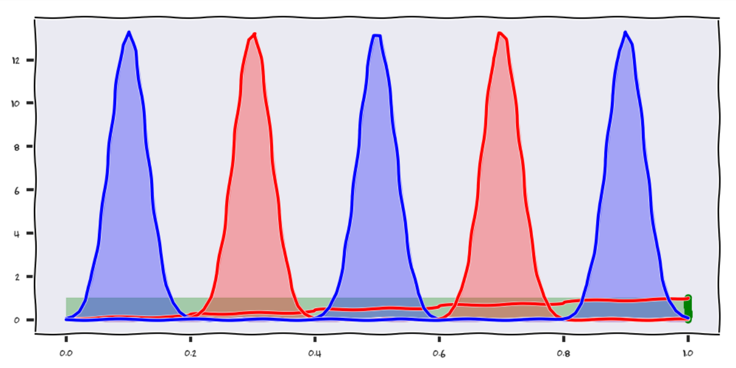
Change of Variables



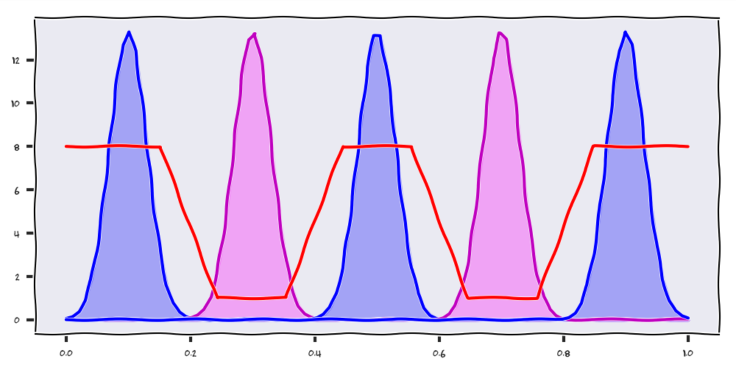
Change of Variables



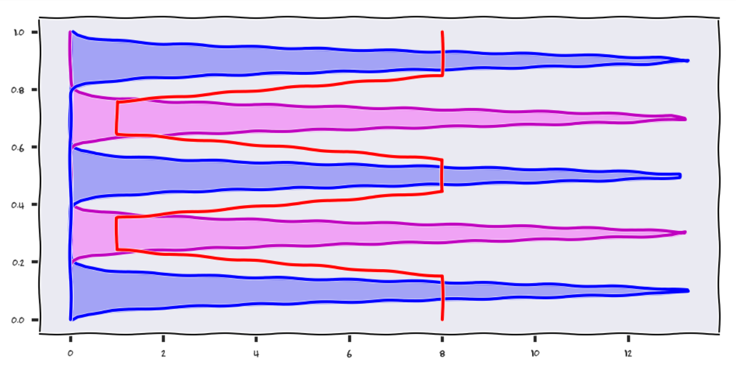
Change of Variables



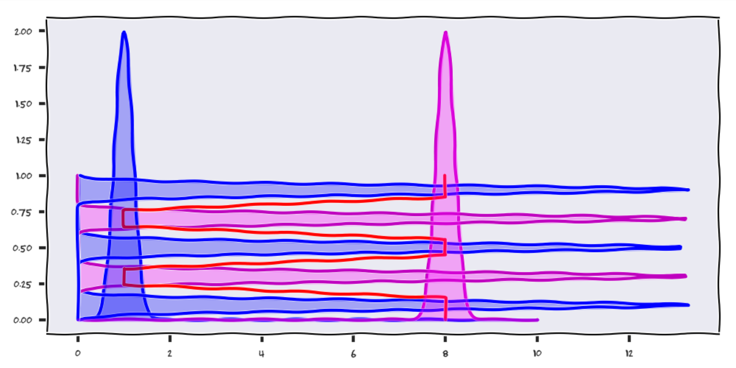
Change of Variables



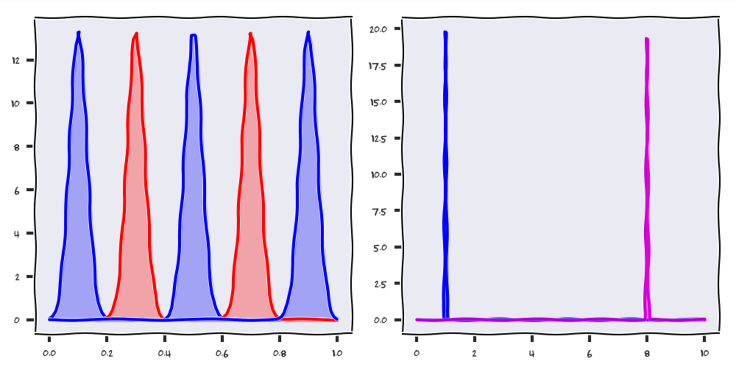
Change of Variables



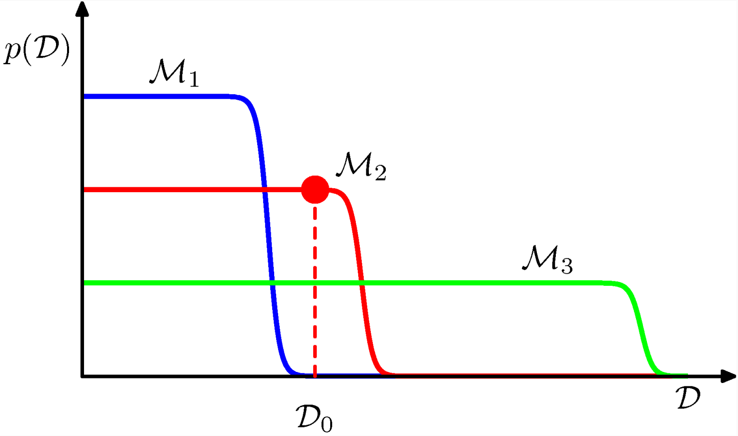
Change of Variables



Change of Variables



MacKay plot



$p(\mathcal{D})$
$\mathcal{M}_1$
$\mathcal{M}_2$
$\mathcal{M}_3$
$\mathcal{D}$
$\mathcal{D}_0$

Composite Functions

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

Learning

Neural Networks

- Specifies a composite function

$$y_i = g(\mathbf{x}_i) = (f_K \circ f_{K-1} \circ \dots_1)(\mathbf{x}_i)$$

- 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)

Neural Networks

- Forward propagation

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

- Forward propagation results in error

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

- Propagate error back to update $\mathbf{W}$

Backpropagation

- "Invented" many times but usually attributed to Rummelhart 1986
- Gradient based updated 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

- Perceptron (no hidden layer)

$$E(\mathbf{W}) = \frac{1}{2}\sum_i \left(y_i - \sum_j w_{ji}x_j\right)^2$$

- Calculate update on first layer

$$\frac{\partial}{\partial w_{ji}}E(\mathbf{W}) = \frac{\partial}{\partial w_{ik}}\frac{1}{2}\sum_i \left(y_i - \sum_j w_{ji}x_j\right)^2$$

Backpropagation

- Perceptron (no hidden layer)

$$E(\mathbf{W}) = \frac{1}{2}\sum_i \left(y_i - \sum_j w_{ji} x_j \right)^2$$

- Calculate update on first layer

$$\frac{\partial}{\partial w_{ji}} E(\mathbf{W}) = \frac{\partial}{\partial w_{ik}} \frac{1}{2}\sum_i \left(y_i - \sum_j w_{ji} x_j \right)^2$$

$$= \frac{1}{2}\sum_i 2(y_i - \underbrace{g(x_i)}_{\text{output of network}}) \frac{\partial}{\partial w_{ji}} \left(y_i - \sum_j w_{ji} x_j \right)$$

Backpropagation

- Perceptron (no hidden layer)

$$E(\mathbf{W}) = \frac{1}{2}\sum_i \left(y_i - \sum_j w_{ji}x_j\right)^2$$

- Calculate update on first layer

$$\frac{\partial}{\partial w_{ji}}E(\mathbf{W}) = \frac{\partial}{\partial w_{ik}}\frac{1}{2}\sum_i \left(y_i - \sum_j w_{ji}x_j\right)^2$$

$$= \frac{1}{2}\sum_i 2(y_i - \underbrace{g(x_i)}_{\text{output of network}})\frac{\partial}{\partial w_{ji}}\left(y_i - \sum_j w_{ji}x_j\right)$$

$$= \sum_i (y_i - g(x_i))(-x_i) = \sum_i (g(x_i) - y_i)x_i$$

Backpropagation

$$E(\mathbf{W}) = \sum_n E_n(\mathbf{W}) = \sum_n \sum_k \frac{1}{2}(g_{nk} - y_{nk})^2$$

$$\frac{\partial E_n}{\partial w_{ji}} = (g_{nj} - y_{nj})x_{ni}$$

- Compute derivative based on a single data-point
- Allows for stochastic updates

Backpropagation: hidden layers

- Activation

$$a_j = \sum_i w_{ji} z_i, \quad z_j = h(a_j)$$

Backpropagation: hidden layers

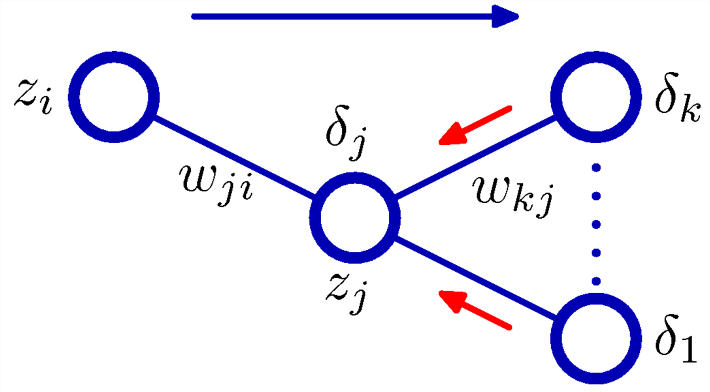
- Activation

$$a_j = \sum_i w_{ji} z_i, \quad z_j = h(a_j)$$

- Derivative

$$\begin{aligned}\frac{\partial E_n}{\partial w_{ji}} &= \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} \\ &= \underbrace{\frac{\partial E_n}{\partial a_j}}_{\delta_j} \frac{\partial}{\partial w_{ji}} \sum_i w_{ji} z_i = \delta_j z_i\end{aligned}$$

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

Automatic Differentiation

- Tensorflow, OpenMX, Theano, Torch, etc.
- Very useful when building composite functions

`log(det(K))`

- Compute log determinant of covariance matrix

log(det(K))

- Compute log determinant of covariance matrix
- Covariance matrix $K \succ 0 \rightarrow$ Cholesky

$$K = L^{\mathrm{T}} L$$

- Compute log determinant of covariance matrix
- Covariance matrix $K \succ 0 \rightarrow$ Cholesky

$$K = L^{\mathrm{T}} L$$

- Simplification

$$\log\left(|K|\right) = \log\left(|L^{\mathrm{T}} L|\right) = \log\left(|L^{\mathrm{T}}| \cdot |L|\right) = 2 \cdot \log|L|$$

`log(det(K))`

- Compute log determinant of covariance matrix
- Covariance matrix $K \succ 0 \rightarrow$ Cholesky

$$K = L^{\mathrm{T}} L$$

- Simplification

$$\log\left(|K|\right) = \log\left(|L^{\mathrm{T}} L|\right) = \log\left(|L^{\mathrm{T}}| \cdot |L|\right) = 2 \cdot \log|L|$$

- L triangular matrix $|A| = \prod \mathrm{diag}(A)$

$$\log\left(|K|\right) = 2 \cdot \log\left(\prod \mathrm{diag}(L)\right) = 2 \sum_{i=1}^{D} \log\left(L_{ii}\right)$$

Architectures

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 Lecunn 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

- Convolution

$$s(x) = \int x(t)w(x - t)\mathrm{d}t$$

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

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

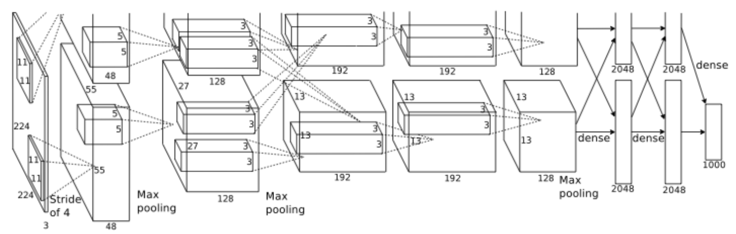
- Makes a lot of sense for images, average nearby pixels

- Pooling

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

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

Convolutional Neural Networks



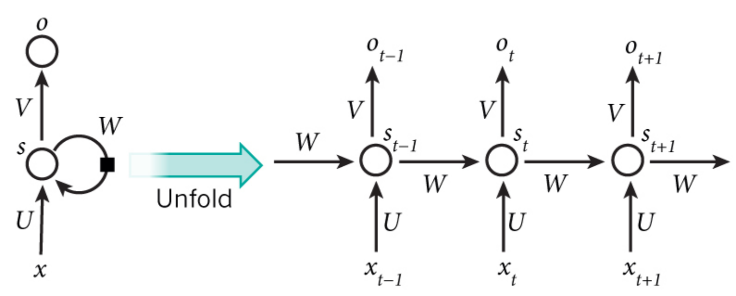
Recurrent Neural Networks

- 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 Networks[2]



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

Architectures

- There exists many many different architectures

[3] Chomsky, 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

[3] Chomsky, 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-justifications[3]

[3] Chomsky, 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-justifications[3]
- They do work really well but don't listen to the pseudo-science dare to have your own intuitions instead

[3] Chomsky, 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

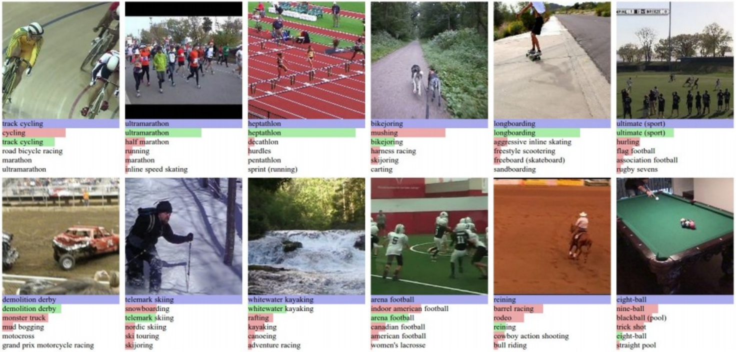
Decision vs. Understanding

- 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

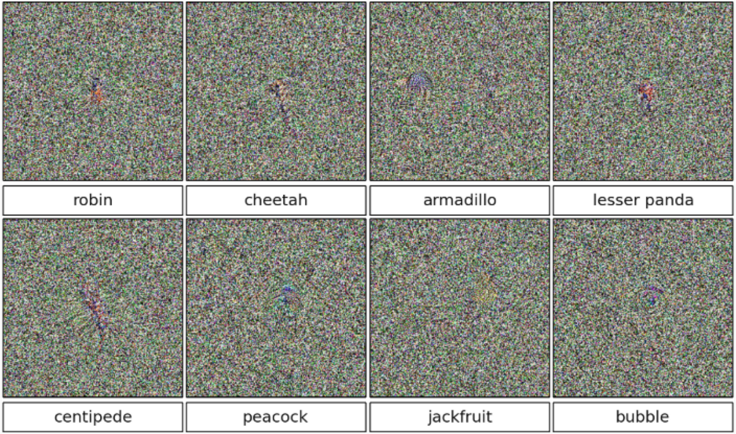
- Neural networks, they way they are described are discriminative this means
  - they do not model the data
  - they do not understand the data
  - there is no such thing as uncertainty
- They are not models they are decision machines
- *Like only revising old exams*

ImageNet



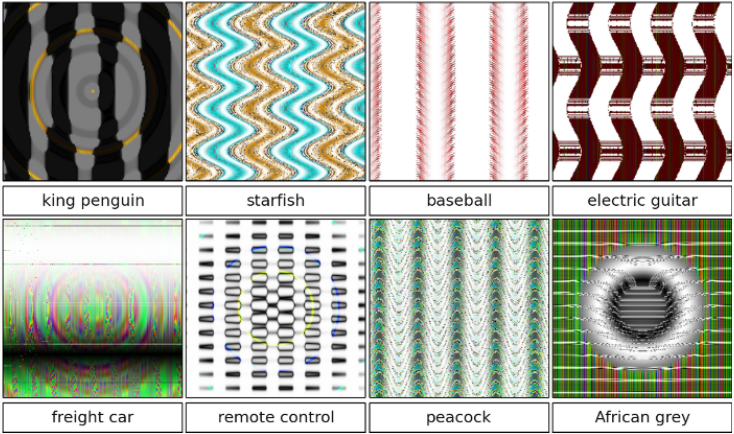
track cycling
cycling
track cycling
road bicycle racing
marathon
ultramarathon
ultramarathon
ultramarathon
half marathon
running
marathon
inline speed skating
heptathlon
heptathlon
decathlon
hurdles
pentathlon
sprint (running)
bikejoring
mushing
bikejoring
harness racing
skijoring
carting
longboarding
longboarding
aggressive inline skating
freestyle scootering
freeboard (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
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
bull riding
eight-ball
nine-ball
blackball (pool)
trick shot
eight-ball
straight pool

Fools[4]



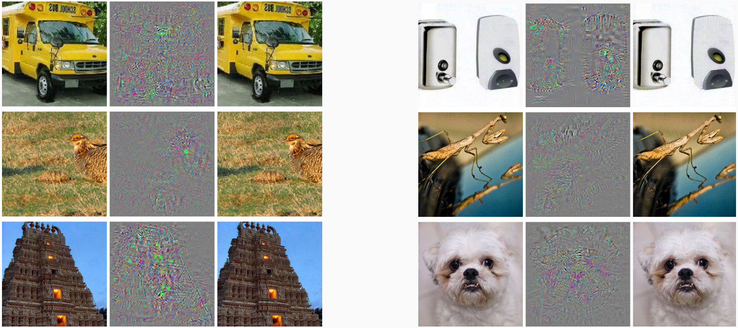
[4] http://www.evolvingai.org/files/DNNsEasilyFooled_cvpr15.pdf

Fools[4]



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

Fools[4]



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

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

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

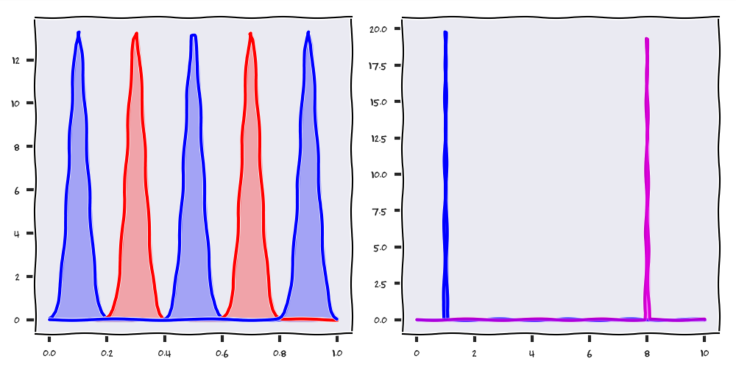
The Black-arts

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

The Black-arts

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

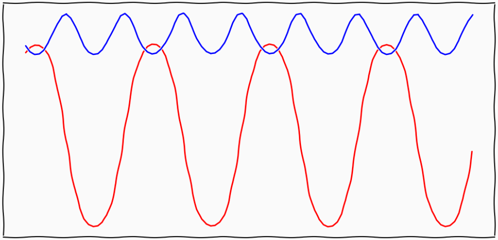
Change of Variables



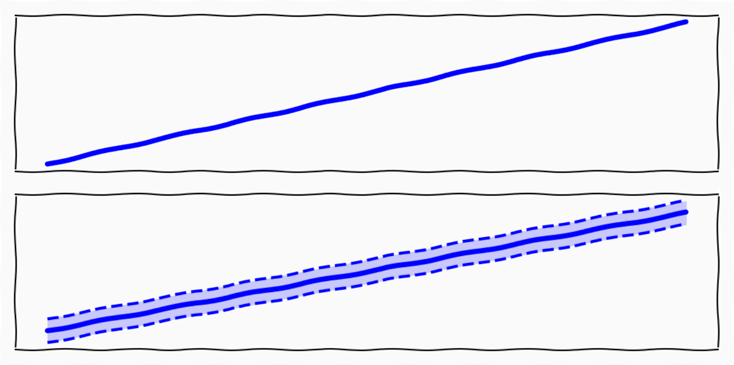
Decision machines

- 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

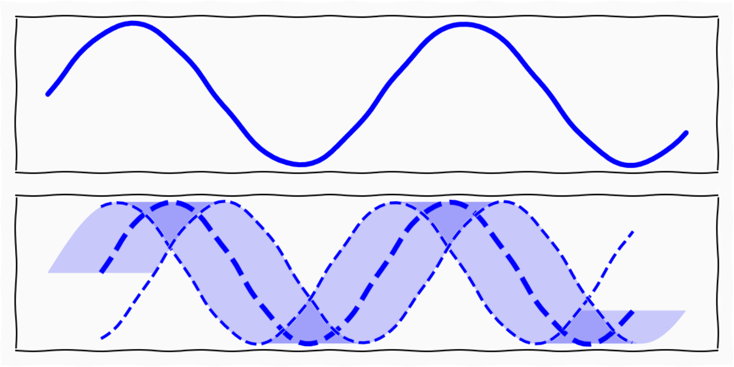
Composition: priors



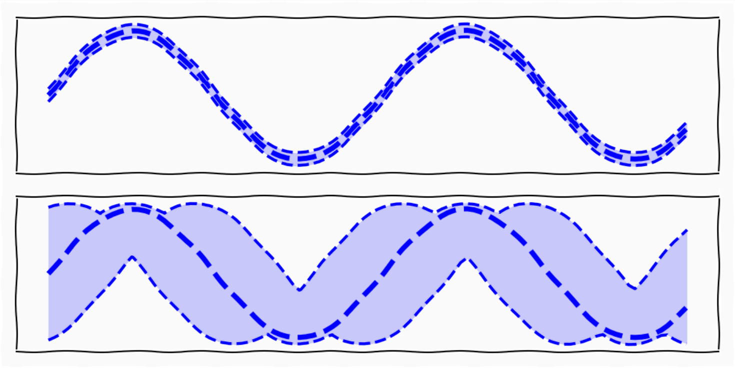
Composition: uncertainty



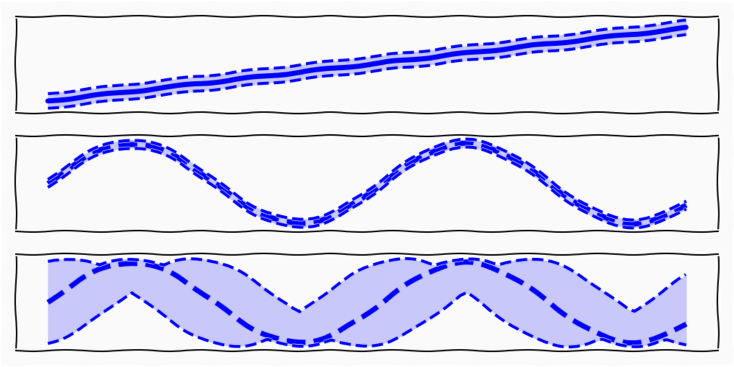
Composition: uncertainty



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