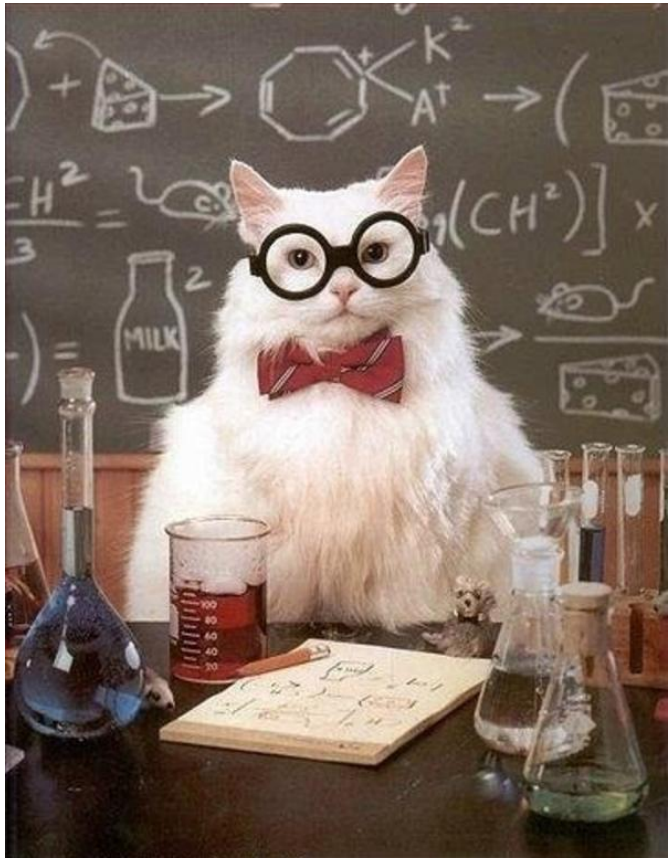
Introduction to Machine Learning feat. TensorFlow

Peter Goldsborough

July 12, 2016

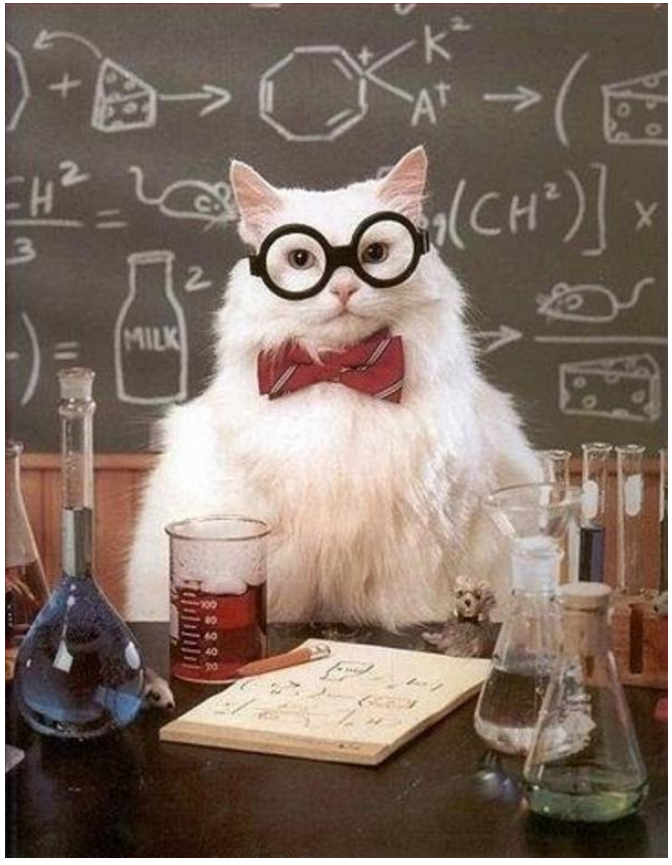
Table of Catents

Table of Catents

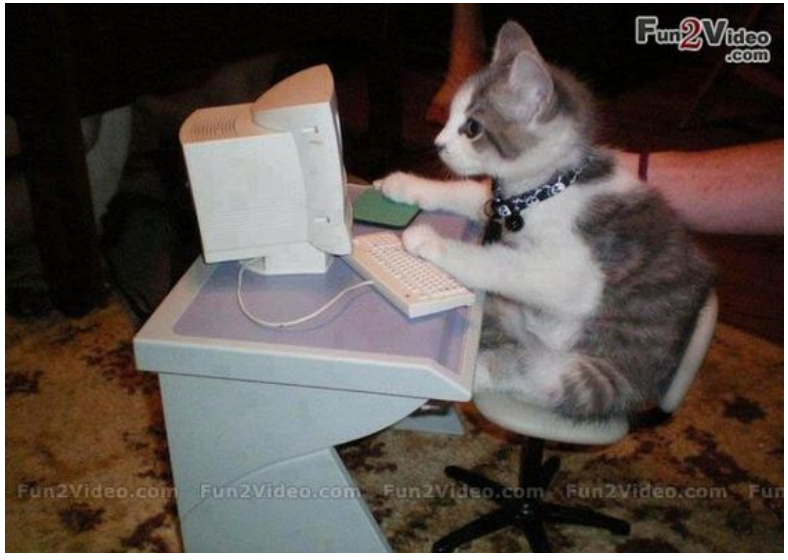


Theory

Table of Catents



Theory



Practice

Background

Background

- CS Student @ TUM

Background

- CS Student @ TUM
- Google & Bloomberg Intern

Background

- CS Student @ TUM
- Google & Bloomberg Intern
- I like cats

Background

- CS Student @ TUM
- Google & Bloomberg Intern
- I like cats

Seminar Topic: *Deep Learning With TensorFlow*

`github.com/peter-can-write/tensorflow-paper`

`github.com/peter-can-talk/python-meetup-munich-2016`

What is Machine Learning?

(and can I eat it?)

Definition I

Machine Learning is cool.

Definition I

Machine Learning is really cool.

Definition II

Machine learning is not magic; it can't get something from nothing.

Definition II

Machine learning is not magic; it can't get something from nothing.

What it does is get more from less.

Definition II

Machine learning is not magic; it can't get something from nothing.

What it does is get more from less.

Learning is like farming, which lets nature do most of the work. Farmers combine seeds with nutrients to grow crops. Learners combine knowledge with data to grow programs.

[Dom12]

Definition III

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

[Mit97]

Definition III

$$f : \text{Image} \to \{\text{cat, banana, spaceship, } \dots\}$$

Definition III

$$f : \text{Image} \to \{\text{cat, banana, spaceship, } \dots\}$$

$$f^\star(x) \approx f(x)$$

The Task, T

The Task, T

- Discriminate by the way an algorithm processes an example $\mathbf{x} \in \mathbb{R}^n$

The Task, T

- Discriminate by the way an algorithm processes an example $\mathbf{x} \in \mathbb{R}^n$
- The output $\mathbf{y}$ can take on various forms

The Task, T

Classification

$$f : \mathbb{R}^n \to \{1, \ldots, k\}$$

The Task, T

Regression

$$f : \mathbb{R}^n \to \mathbb{R}$$

Experience E

How do we make our algorithm learn?

Experience E

How do we make our algorithm learn?

Supervised Learning

Experience E

How do we make our algorithm learn?

Supervised Learning

Unsupervised Learning

How do I Machine Learning?

Methods

Methods

Methods

$$\mathbf{b} = (R, G, B)^\top$$

Methods

$$\mathbf{b} = (R, G, B)^\top$$

$$f : \mathbf{b} \mapsto \text{market price} \in \mathbb{R}$$

Methods: Features

Methods: Features

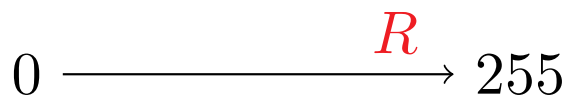
- Components of $\mathbf{b}$ are **features**

Methods: Features

- Components of $\mathbf{b}$ are **features**
- Each feature represents one axis in space

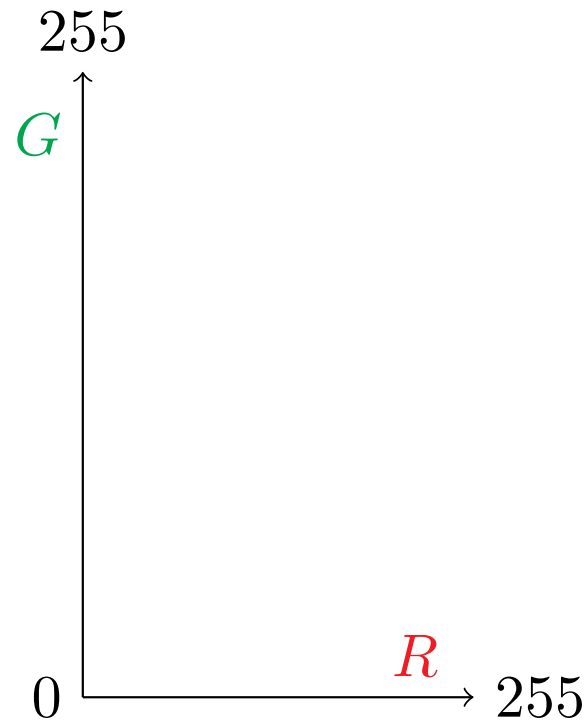
Methods: Features

- Components of $\mathbf{b}$ are **features**
- Each feature represents one axis in space



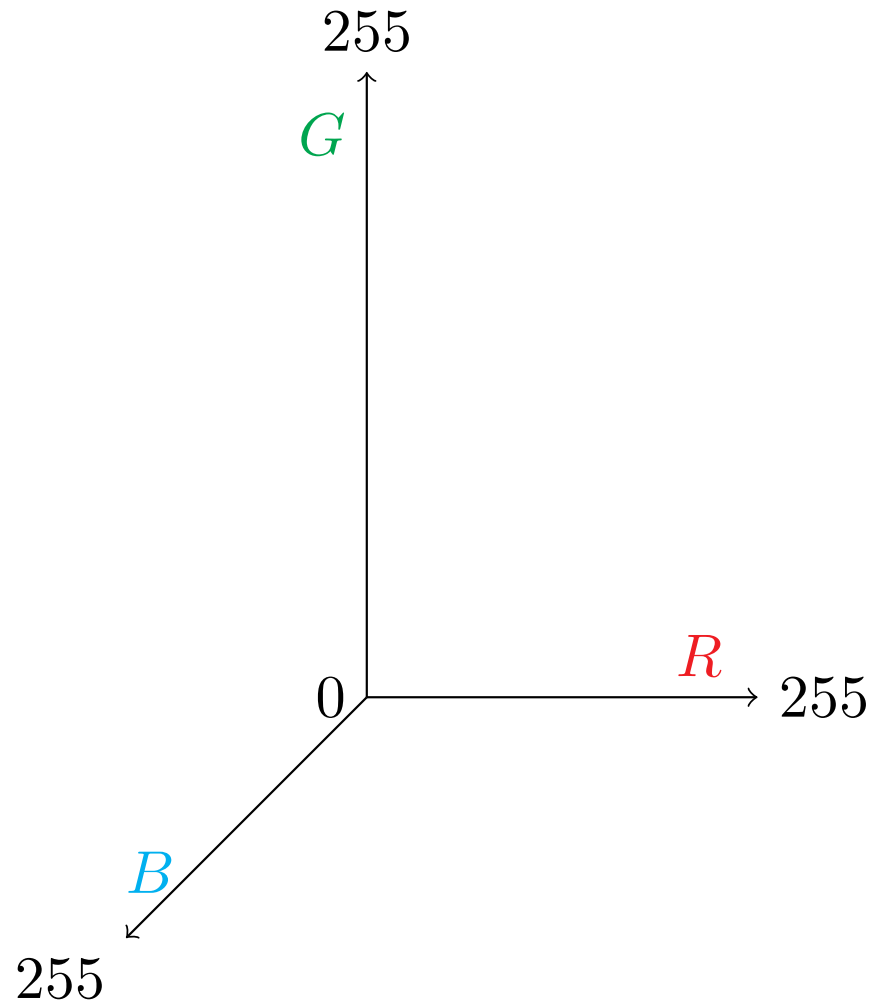
Methods: Features

- Components of $\mathbf{b}$ are **features**
- Each feature represents one axis in space



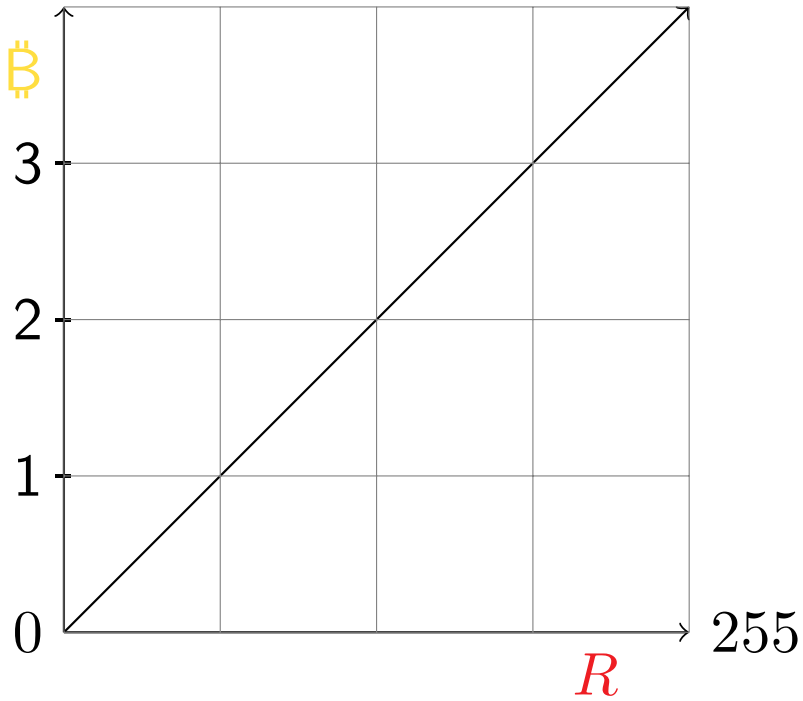
Methods: Features

- Components of $\mathbf{b}$ are **features**
- Each feature represents one axis in space



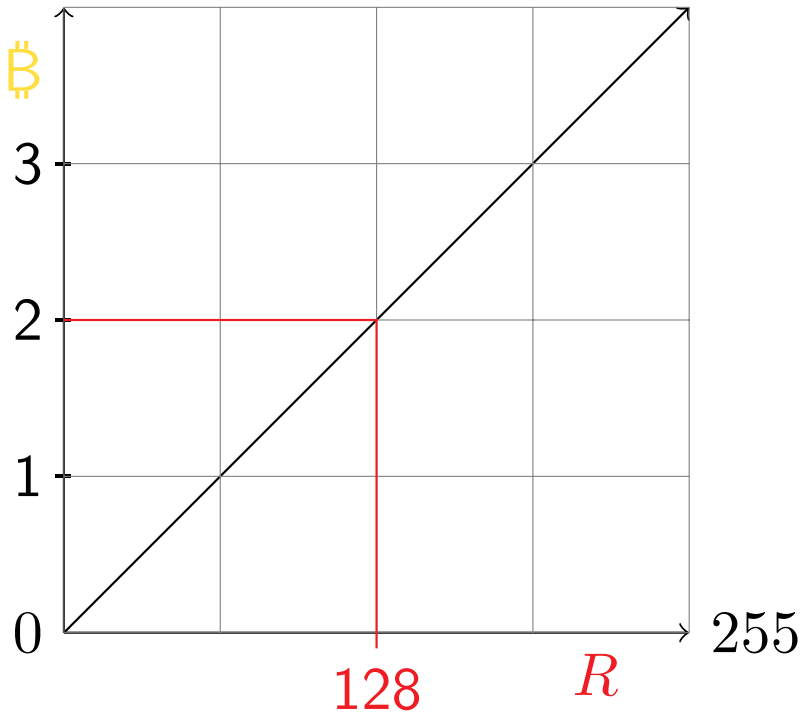
Methods: Features

- The components of each vector $\mathbf{b}$ are our **features**
- Each feature represents one axis in space



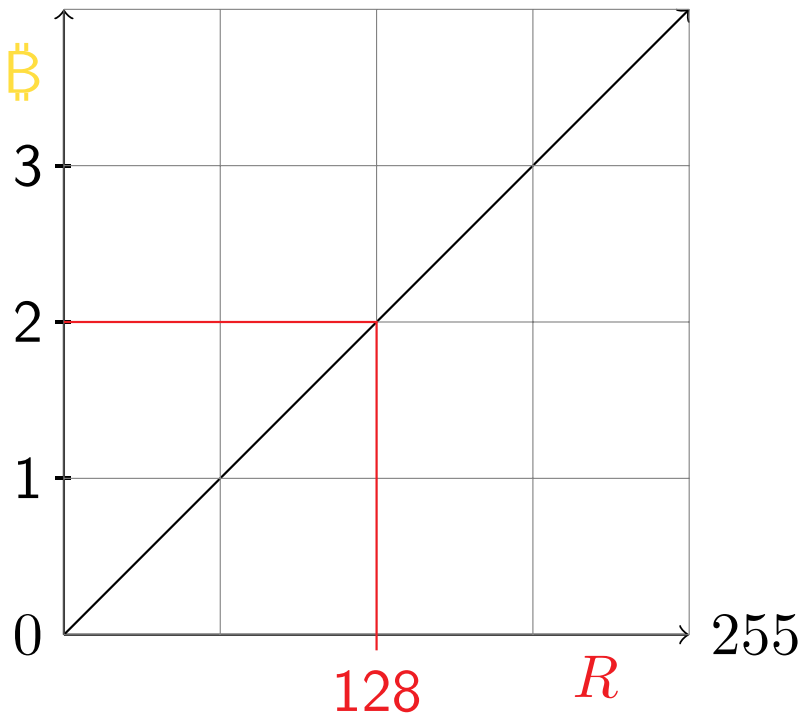
Methods: Features

- The components of each vector $\mathbf{b}$ are our **features**
- Each feature represents one axis in space



Methods: Features

- The components of each vector $\mathbf{b}$ are our **features**
- Each feature represents one axis in space
- Each feature should contribute to some extent to the output value



Methods: Features

- We weight each component

$$f(\mathbf{b}) = w_1 b_1 + w_2 b_2 + w_3 b_3$$

Methods: Features

- We weight each component
- We add a bias c as an offset

$$f(\mathbf{b}) = w_1 b_1 + w_2 b_2 + w_3 b_3 + c$$

Methods: Features

- We weight each component
- We add a bias c as an offset
- We expect our algorithm to learn $\mathbf{w}$ and c

$$f(\mathbf{b}) = \mathbf{w}^\top \mathbf{b} + c$$

Methods: Data

- To make our algorithm learn, we need to feed it (lots of) data

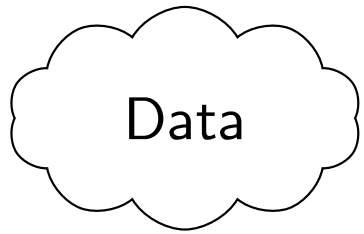
Methods: Data

- To make our algorithm learn, we need to feed it (lots of) data

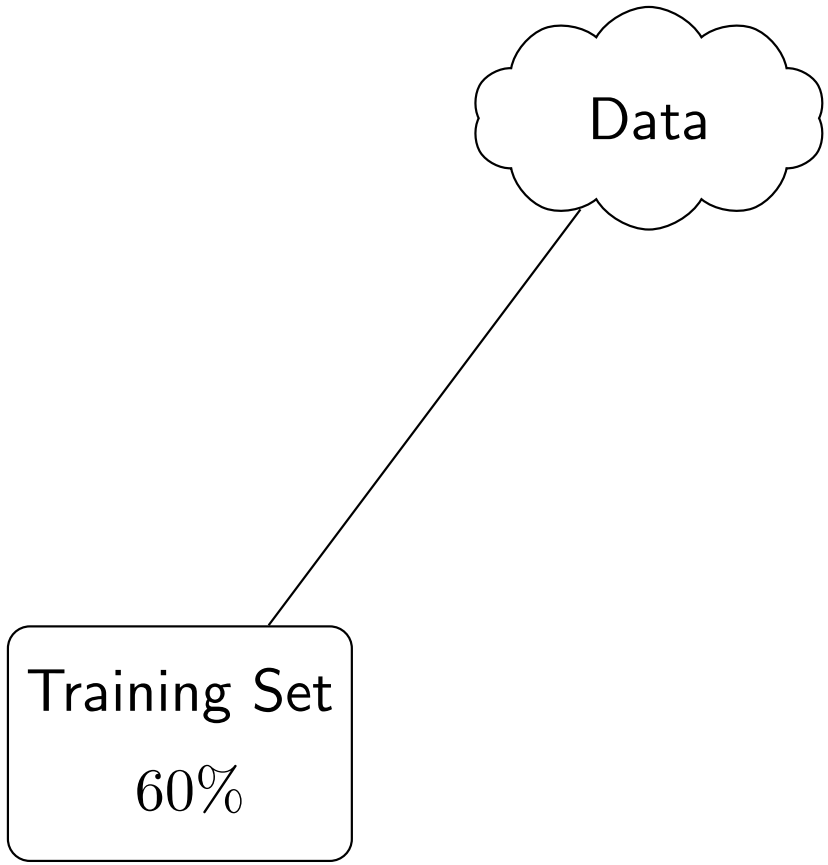
$$
\begin{array}{c} \\ n \end{array}
\begin{array}{c}
\begin{array}{ccc} R & G & B \end{array} \\
\begin{bmatrix} 34 & 147 & 73 \\ 247 & 69 & 13 \\ 66 & 66 & 66 \end{bmatrix}
\end{array}
$$

Design Matrix

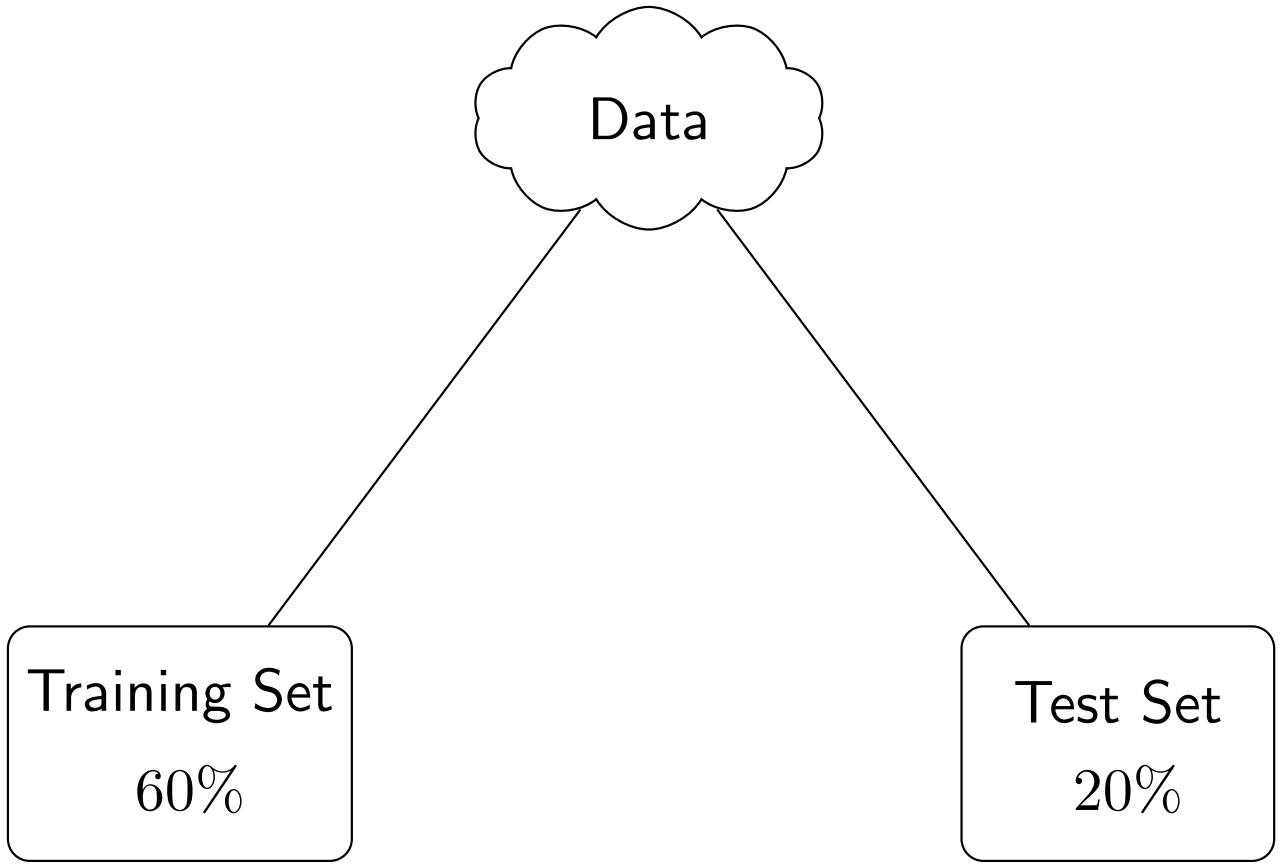
Methods: Data



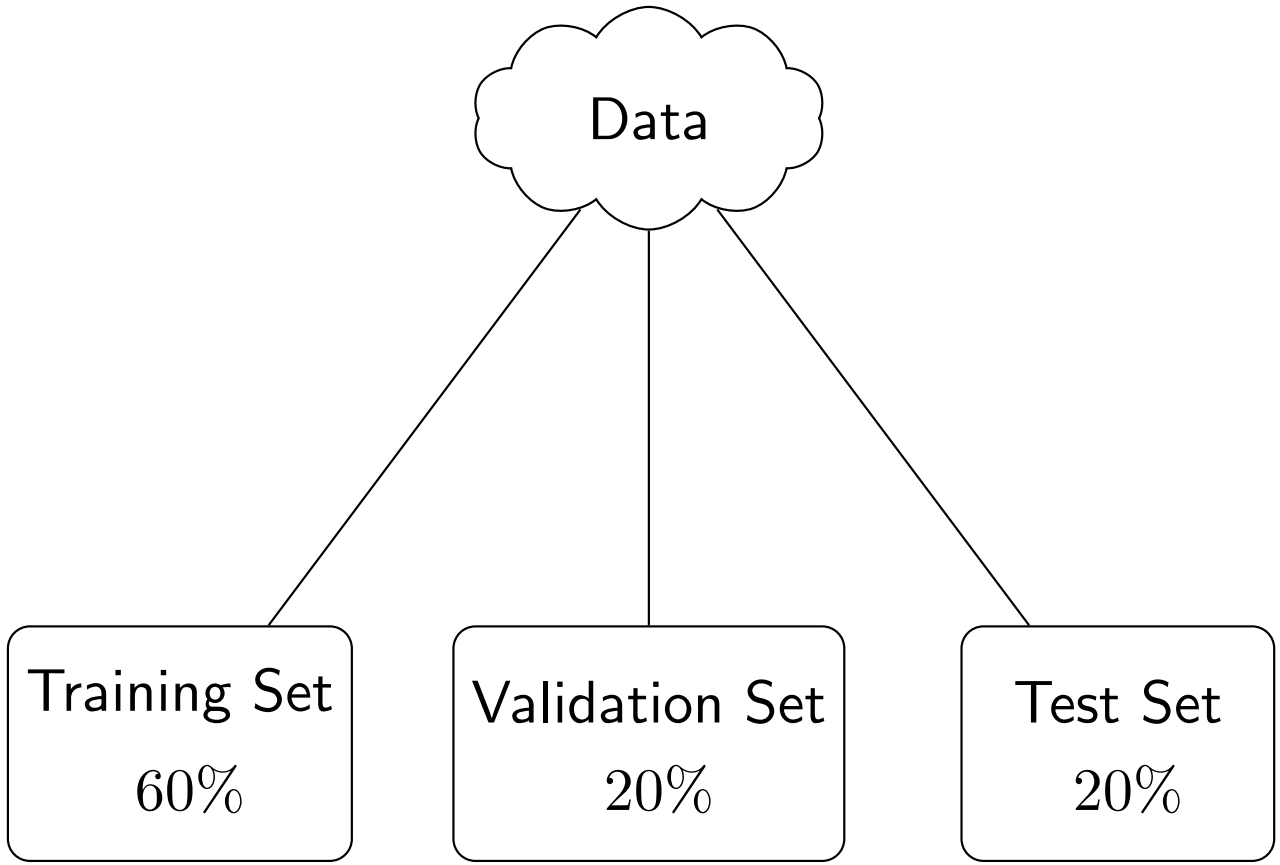
Methods: Data



Methods: Data



Methods: Data



Methods: Measuring Performance

- Need a way to evaluate the performance of our algorithm (P)

Methods: Measuring Performance

- Need a way to evaluate the performance of our algorithm (P)
- Once we know how well it is performing, we can update it

Methods: Measuring Performance

- Need a way to evaluate the performance of our algorithm (P)
- Once we know how well it is performing, we can update it

$$L(\mathbf{y}, \hat{\mathbf{y}}) \in \mathbb{R}$$

Methods: Measuring Performance

Regression

Methods: Measuring Performance

Regression

$$\underbrace{\begin{bmatrix} 34 & 147 & 73 \\ 247 & 69 & 13 \\ 66 & 66 & 66 \end{bmatrix}}_{\mathbf{D}} \times \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}}_{\mathbf{w}} +_b c = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}}_{\mathbf{y}} \xrightarrow{Target} \underbrace{\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix}}_{\hat{\mathbf{y}}}$$

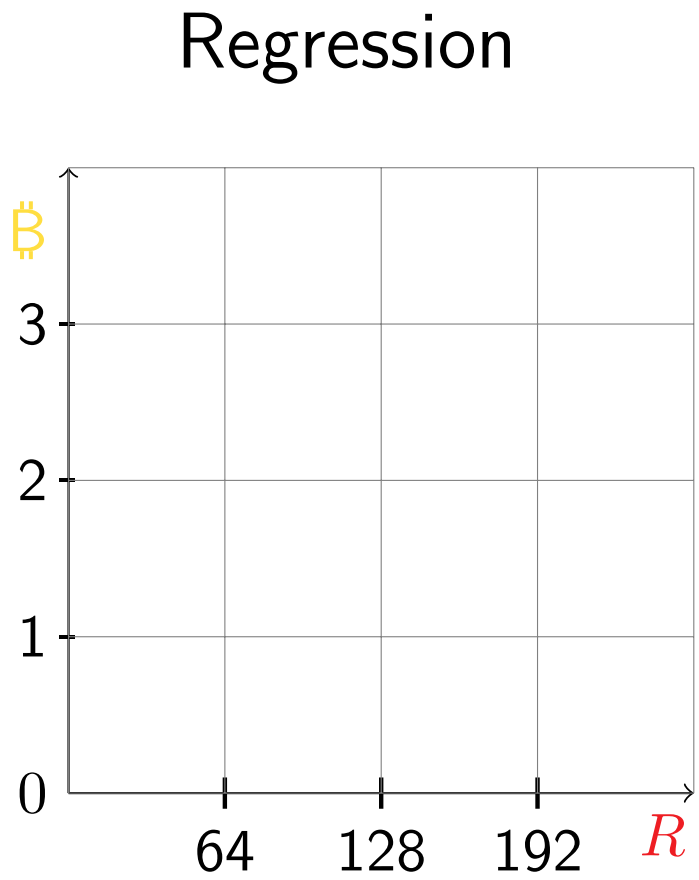
Methods: Measuring Performance

Regression

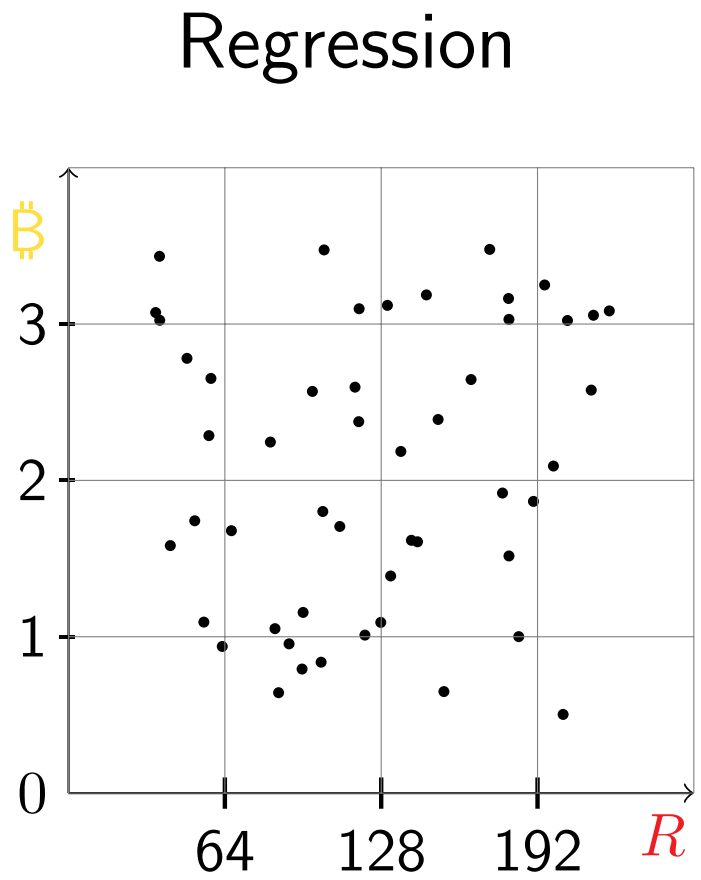
$$\underbrace{\begin{bmatrix} 34 & 147 & 73 \\ 247 & 69 & 13 \\ 66 & 66 & 66 \end{bmatrix}}_{\mathbf{D}} \times \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}}_{\mathbf{w}} +_b c = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}}_{\mathbf{y}} \xrightarrow{Target} \underbrace{\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix}}_{\hat{\mathbf{y}}}$$

$$L(\mathbf{y}, \hat{\mathbf{y}}) = MSE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Methods: Measuring Performance

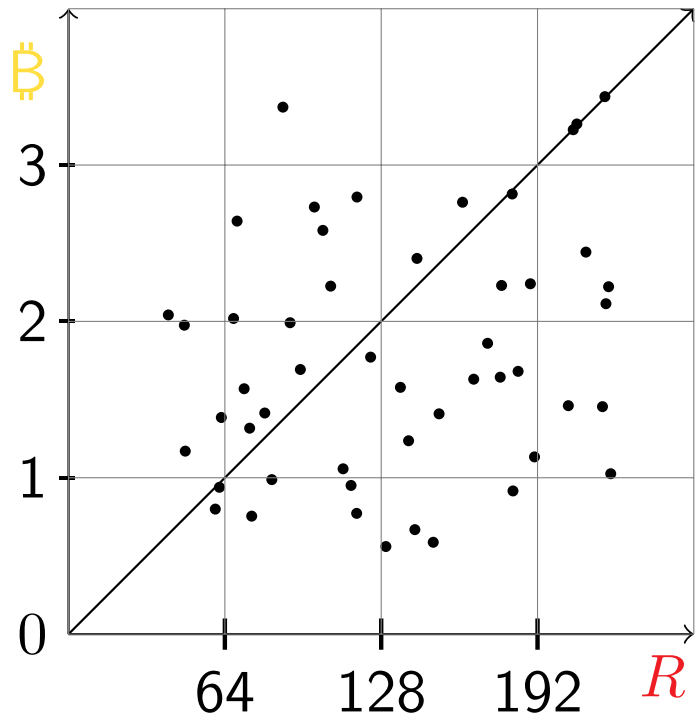


Methods: Measuring Performance



Methods: Measuring Performance

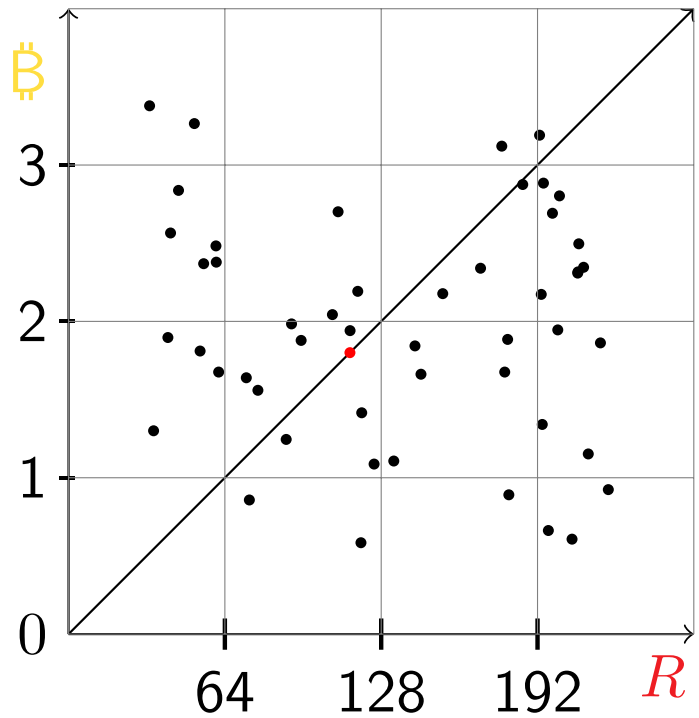
Regression



$$f(\mathbf{b}) = b_1 + b_2 + b_3$$

Methods: Measuring Performance

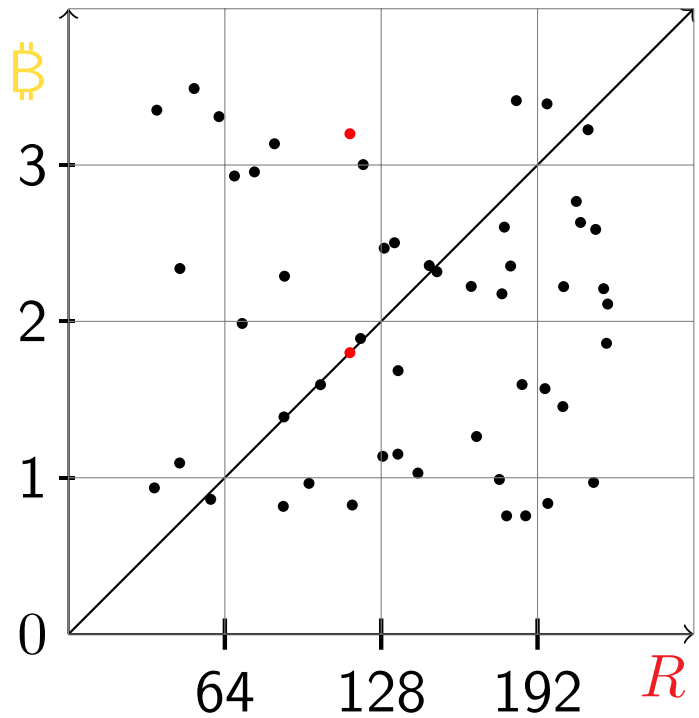
Regression



$$f(\mathbf{b}) = b_1 + b_2 + b_3$$

Methods: Measuring Performance

Regression



$$f(\mathbf{b}) = b_1 + b_2 + b_3$$

Methods: Gradient Descent

- Through $L(\mathbf{y}, \hat{\mathbf{y}})$ we know how our algorithm is performing
- We can compute its rate of change (derivative)
- Move our weights in the opposite direction

Methods: Gradient Descent

- Through $L(\mathbf{y}, \hat{\mathbf{y}})$ we know how our algorithm is performing
- We can compute its rate of change (derivative)
- Move our weights in the opposite direction

This is the core idea behind Supervised Learning

Methods: Gradient Descent

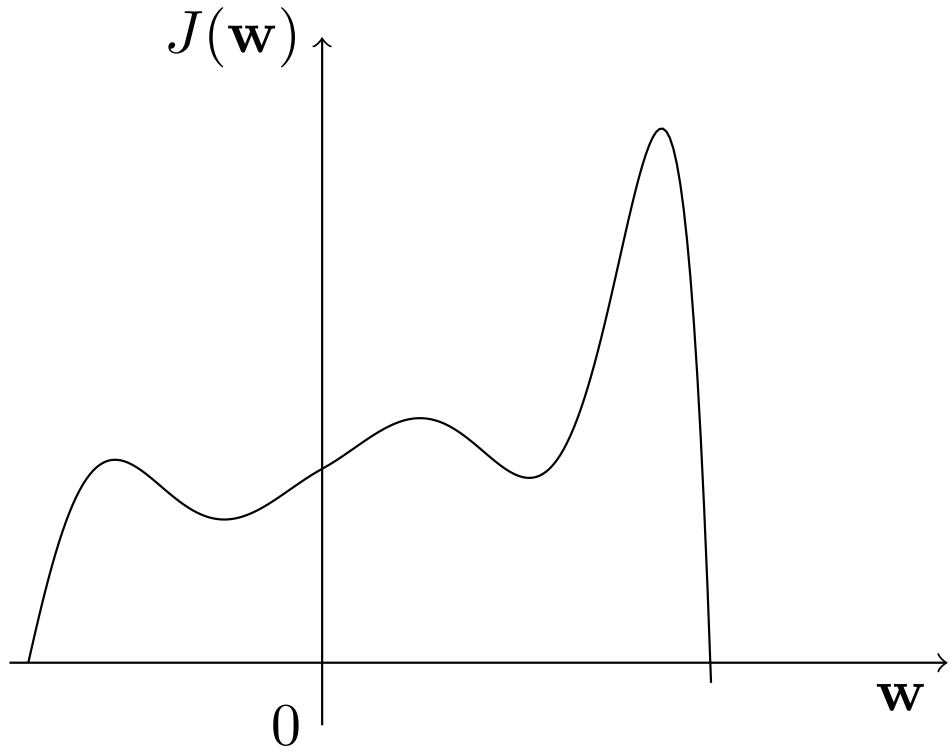
- $L(\mathbf{y}, \hat{\mathbf{y}})$ depends on $\mathbf{y}$ and $\hat{\mathbf{y}}$, but *also* $\mathbf{w}$: $L(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w})$

Methods: Gradient Descent

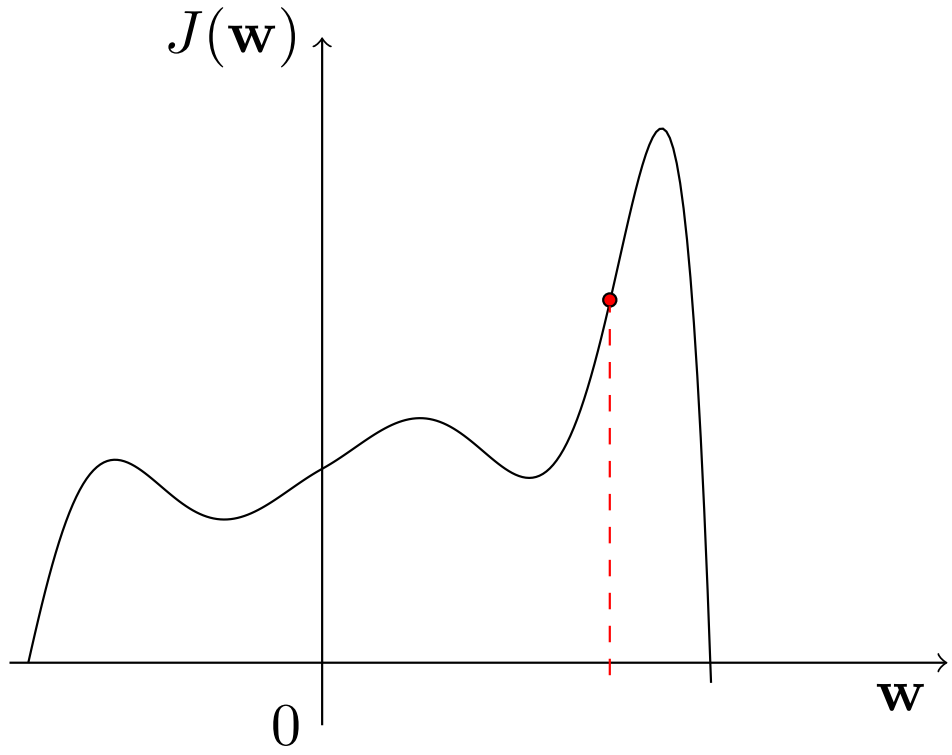
- $L(\mathbf{y}, \hat{\mathbf{y}})$ depends on $\mathbf{y}$ and $\hat{\mathbf{y}}$, but *also* $\mathbf{w}$: $L(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w})$
- We can think of $\mathbf{y}, \hat{\mathbf{y}}$ as being *constant* and write:

$$J(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{y}_i, \hat{\mathbf{y}}_i; \mathbf{w})$$

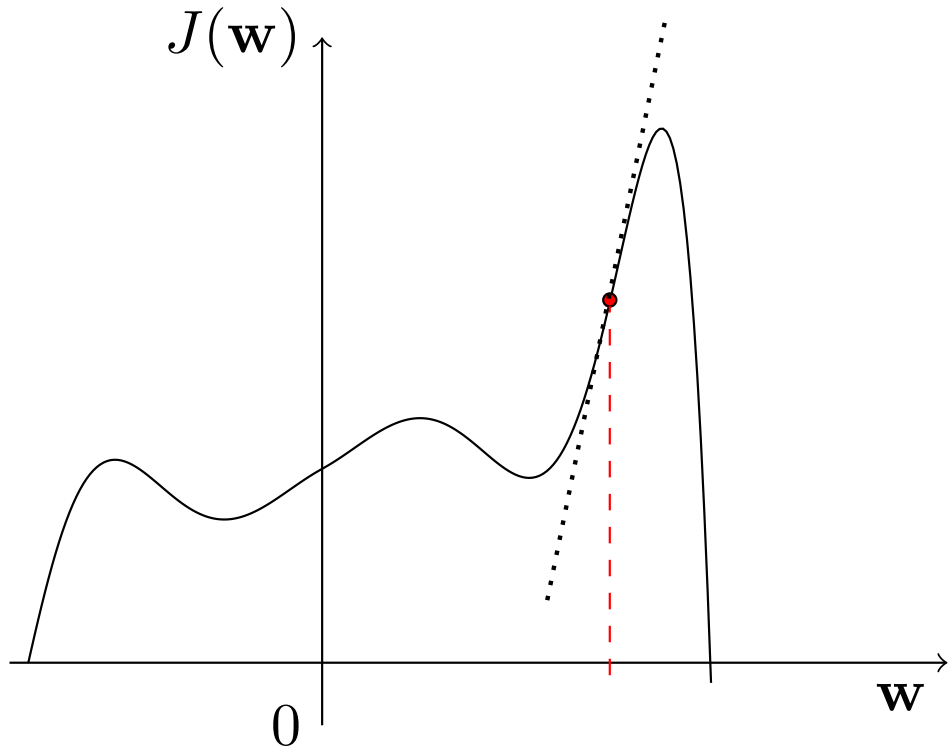
Methods: Gradient Descent



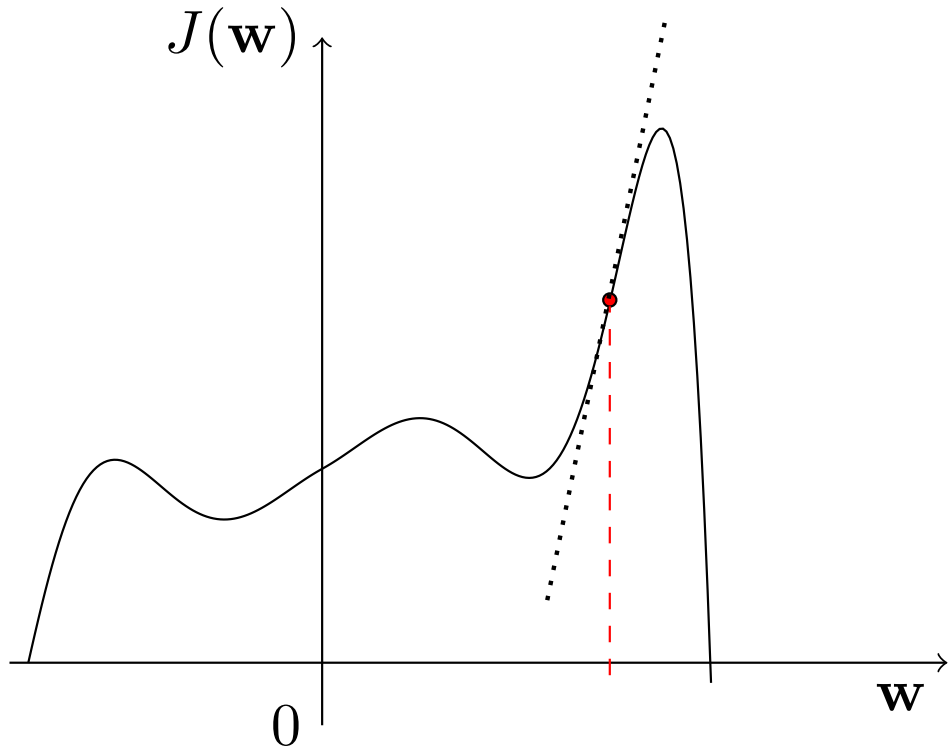
Methods: Gradient Descent



Methods: Gradient Descent



Methods: Gradient Descent



$\mathbf{w} \leftarrow \mathbf{w} -$ rate of change at $\mathbf{w}$

Methods: Regularization

- We want our models to *generalize*

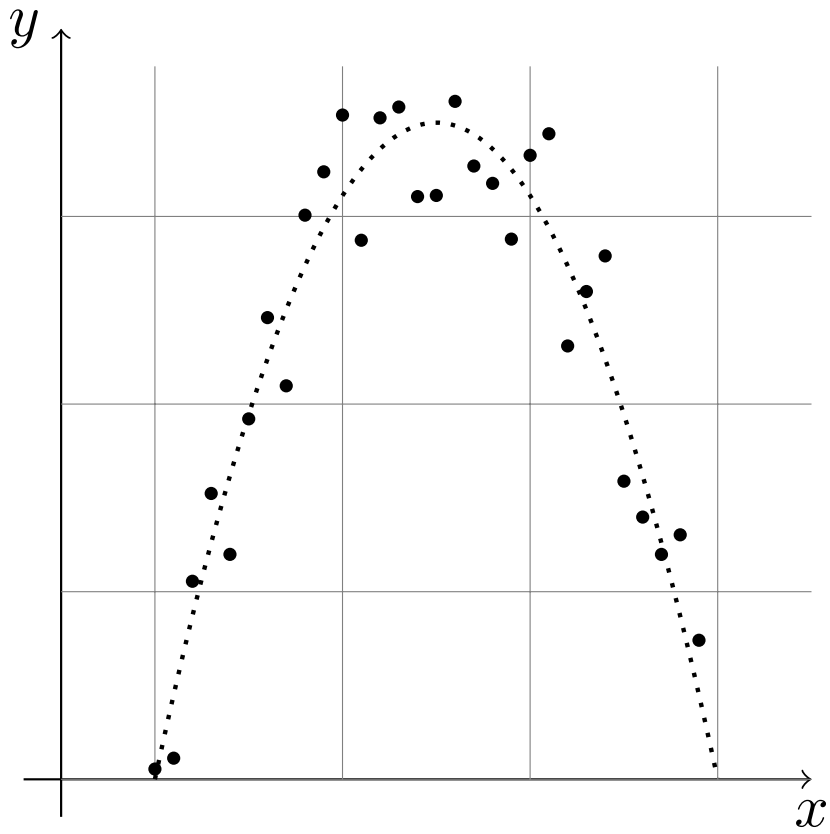
Methods: Regularization

- We want our models to *generalize*
- *Regularization* helps us with this

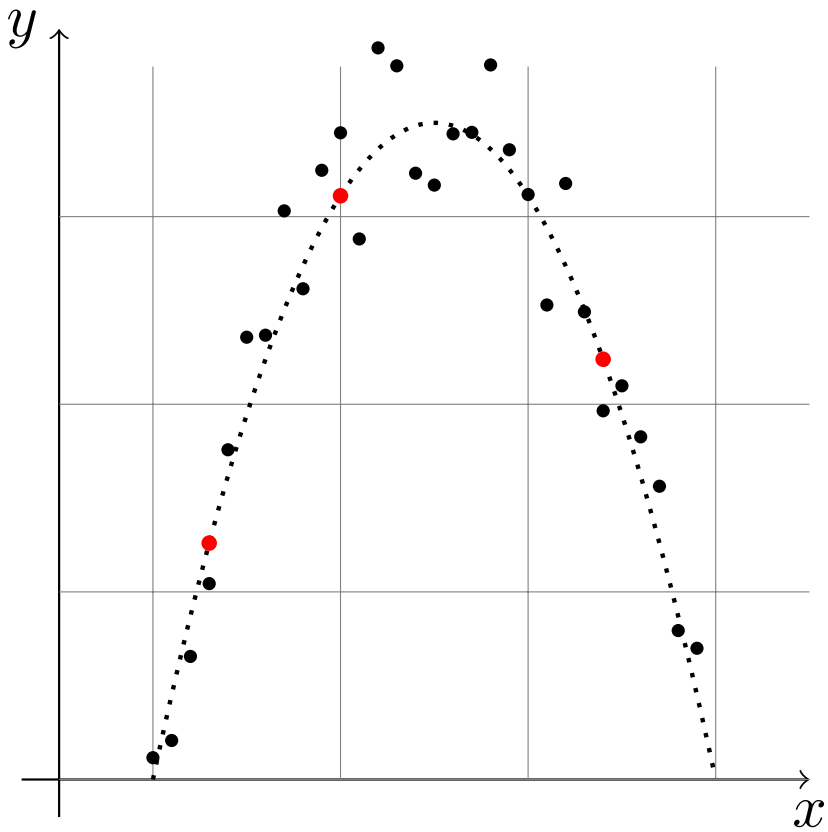
Methods: Regularization

- We want our models to *generalize*
- *Regularization* helps us with this
- We try to influence a model's *capacity*

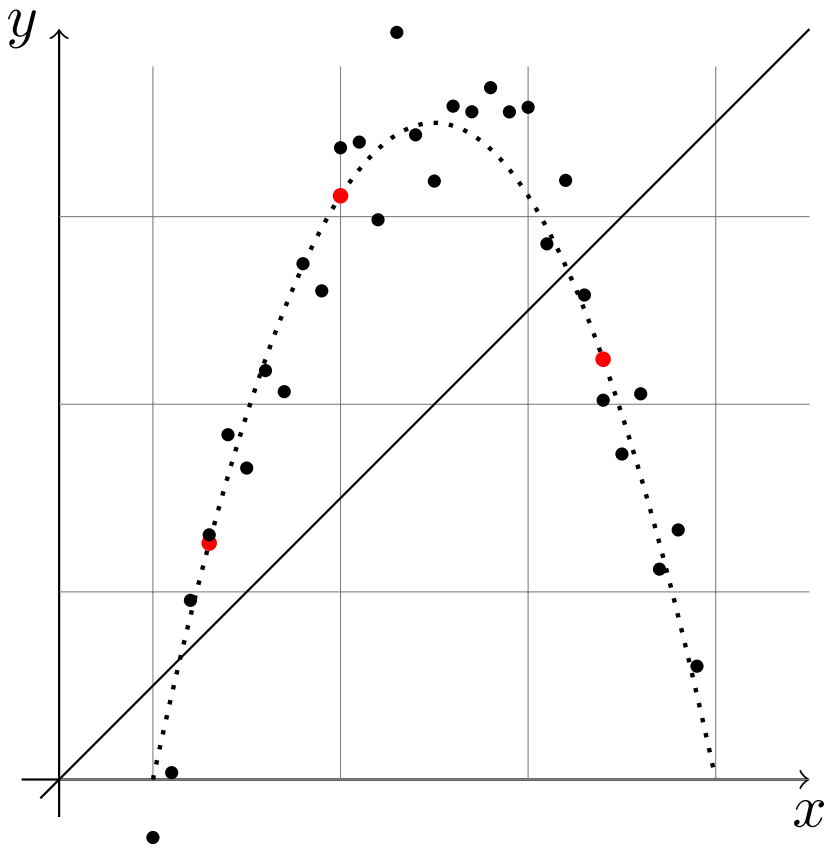
Concepts: Model Capacity



Concepts: Model Capacity

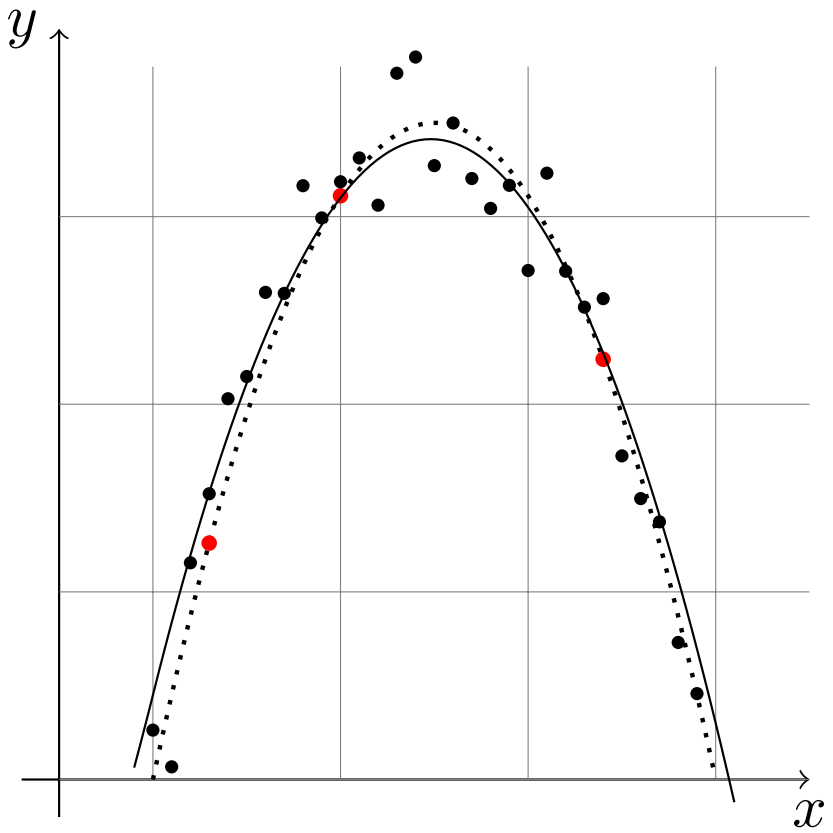


Concepts: Model Capacity



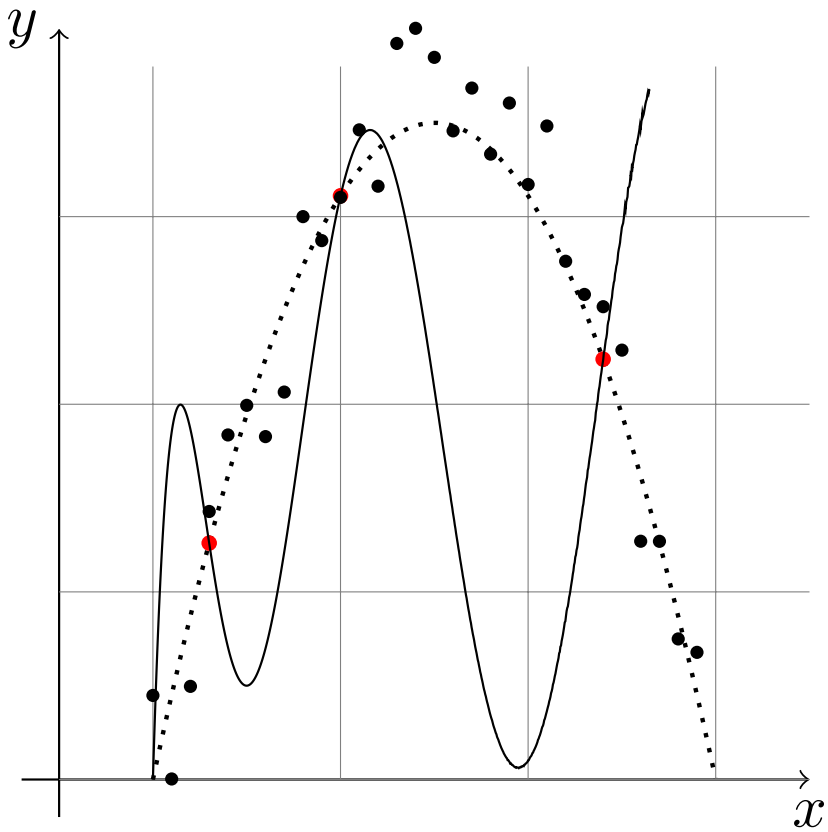
Underfitting

Concepts: Model Capacity



Just Right

Concepts: Model Capacity



Overfitting

Methods: Regularization

$$
\begin{aligned}
f(x) = & \, 0x^8 + 0x^7 + 0x^6 \\
& + 0x^5 + 0x^4 + 0x^3 \\
& - 1.56x^2 + 4.67x + 0
\end{aligned}
$$

$$
\begin{aligned}
g(x) = & -0.69x^8 + 10.90x^7 - 69.52x^6 \\
& + 229.12x^5 - 413.77x^4 + 399.50x^3 \\
& - 185.95x^2 + 33.51x + 7.17 \cdot 10^{-8}
\end{aligned}
$$

Methods: Regularization

$$J(\mathbf{w}) = MSE(\mathbf{y}, \hat{\mathbf{y}}) + \lambda(\mathbf{w}^\top \mathbf{w})$$

Time to go Deeper

What if the meaning of life is to spend your time thinking about the meaning of life?

Deep Learning

Deep Learning

The why

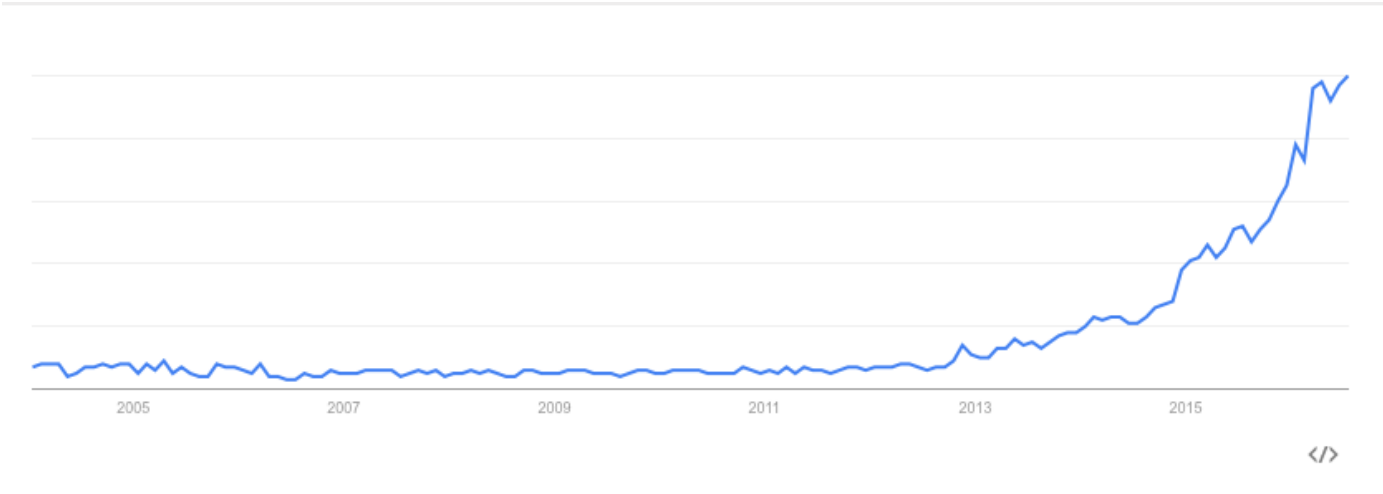
Deep Learning

The why, the what

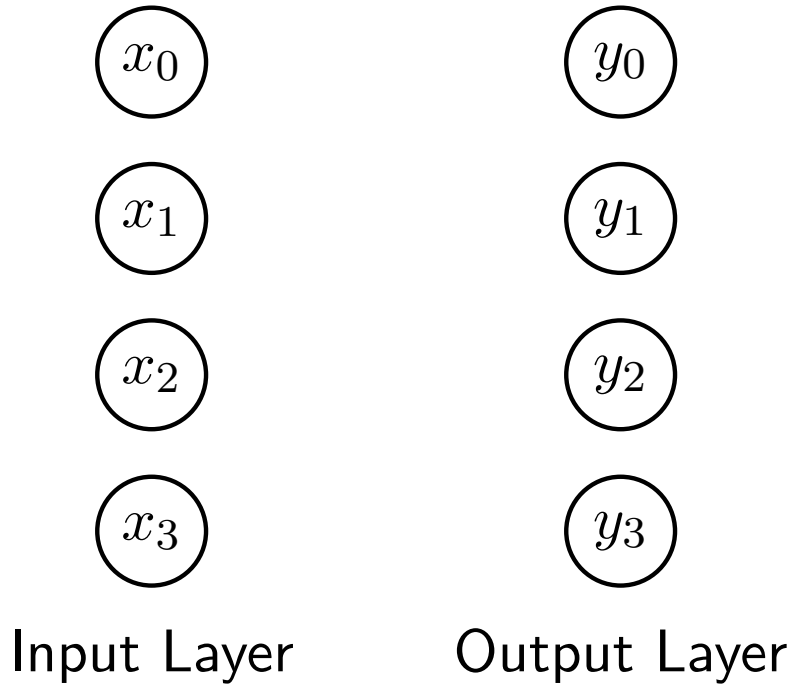
Deep Learning

The why, the what and the ugly.

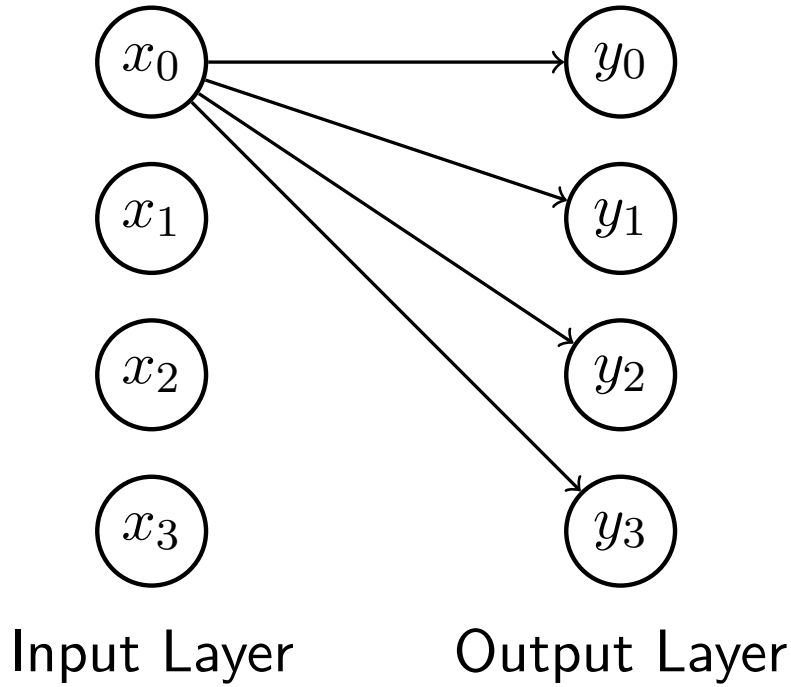
Deep Learning



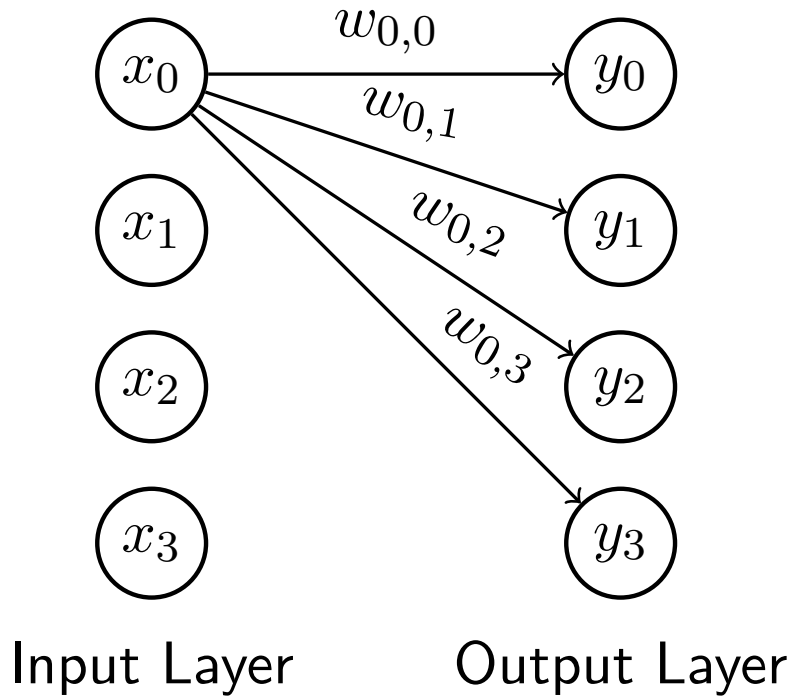
Neural Networks



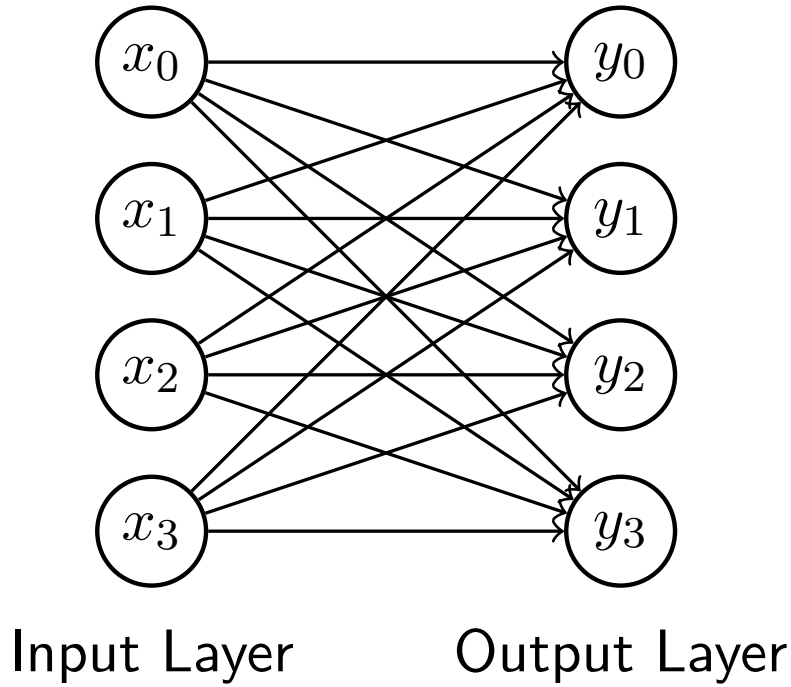
Neural Networks



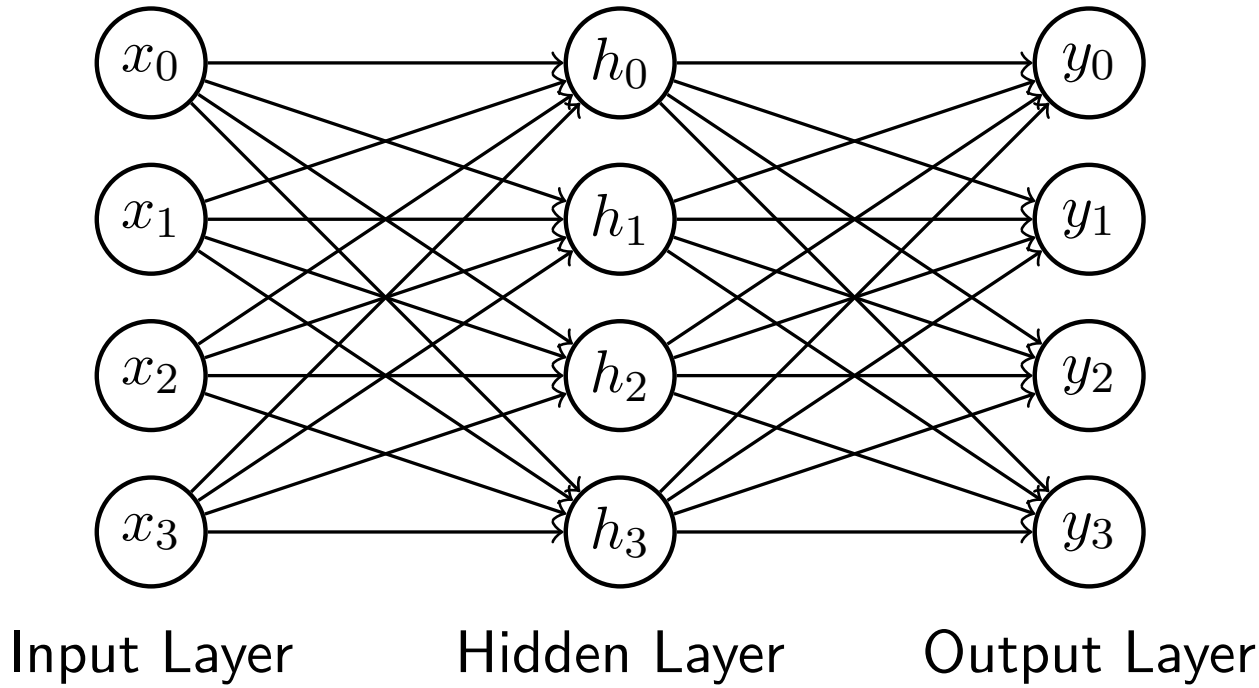
Neural Networks



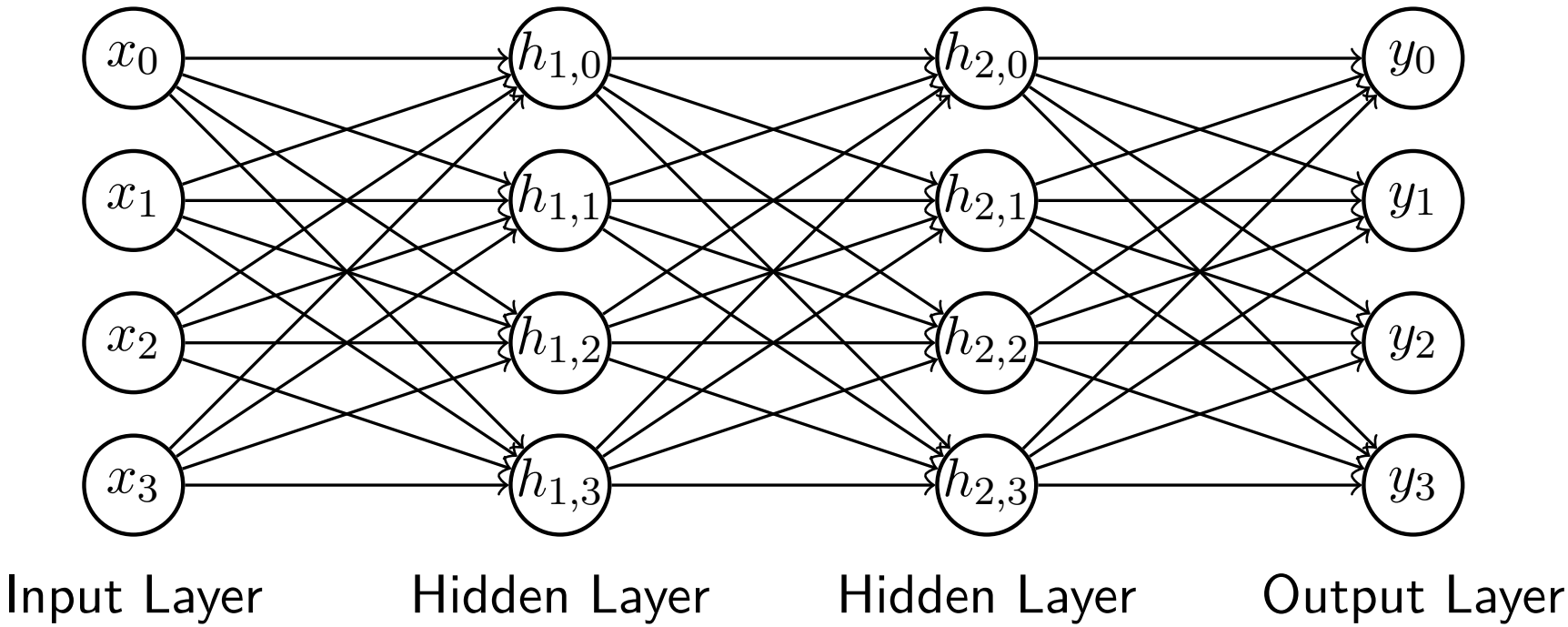
Neural Networks



Deep Neural Networks



Deep Neural Networks



Deep Learning

Deep Learning assumes that data is structured

Deep Learning assumes that data is structured

hierarchically

Convolutional Neural Networks

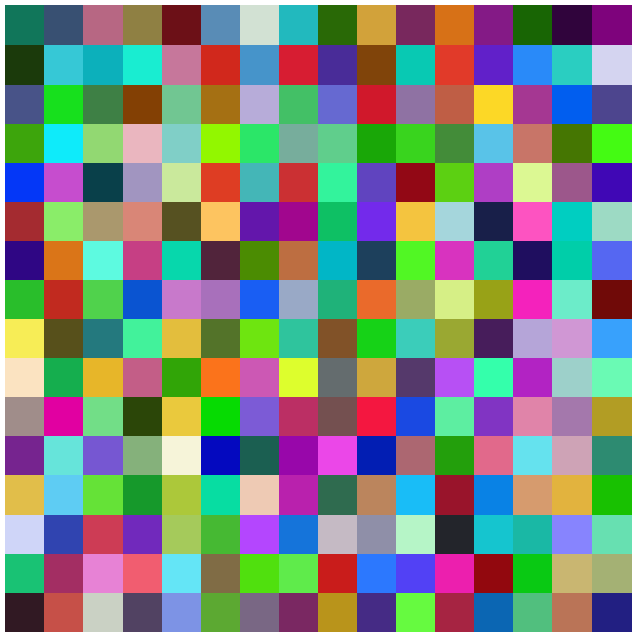
- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition

Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images

Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



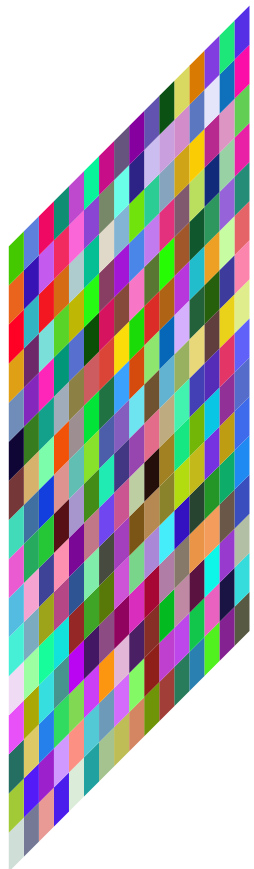
Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images

5	34	18	95	91	46	46	64	30	39	48	67	49	39	85	13
15	11	61	19	61	40	75	97	47	40	56	17	4	64	39	65
2	31	20	44	52	93	43	20	75	93	46	76	69	61	3	72
52	17	58	62	19	26	23	48	86	91	3	71	98	33	14	71
48	31	81	67	41	22	81	95	37	83	88	89	29	14	70	84
6	32	2	56	7	19	43	70	73	79	0	1	23	11	92	28
28	59	39	44	45	13	29	76	87	99	46	77	34	26	37	4
25	34	74	83	4	75	84	36	82	25	23	78	59	40	1	24
79	44	64	9	89	30	43	23	36	40	96	71	88	14	4	84
12	8	72	87	65	59	83	47	51	93	30	60	67	19	82	32
70	24	38	32	22	68	56	45	68	45	84	27	1	76	84	68
2	97	93	5	76	8	5	59	61	74	82	22	41	97	75	60
44	36	21	39	30	35	56	26	36	81	71	18	71	72	20	2
48	99	58	65	29	88	29	93	62	60	62	72	26	35	60	39
57	37	23	83	93	3	54	69	8	52	2	36	48	84	96	85
23	35	88	80	53	75	33	63	20	57	22	17	47	29	22	65

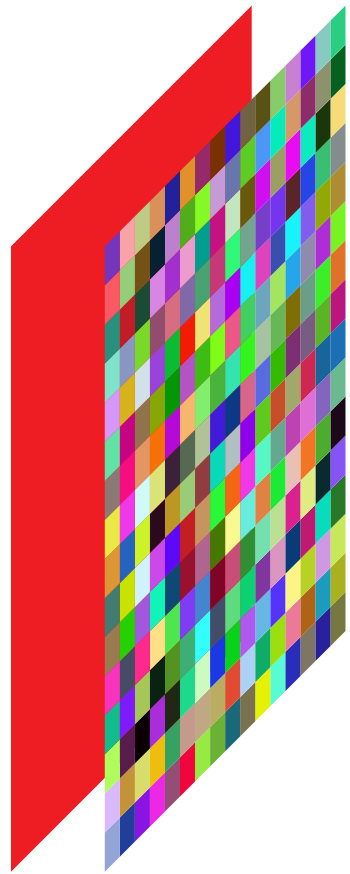
Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



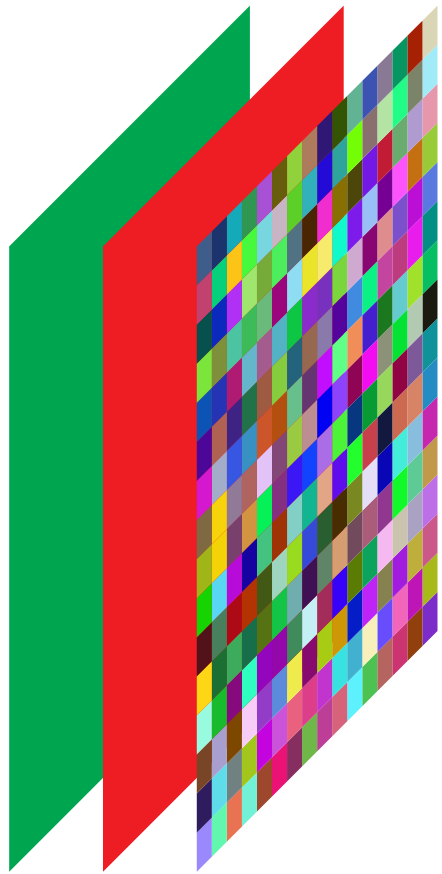
Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



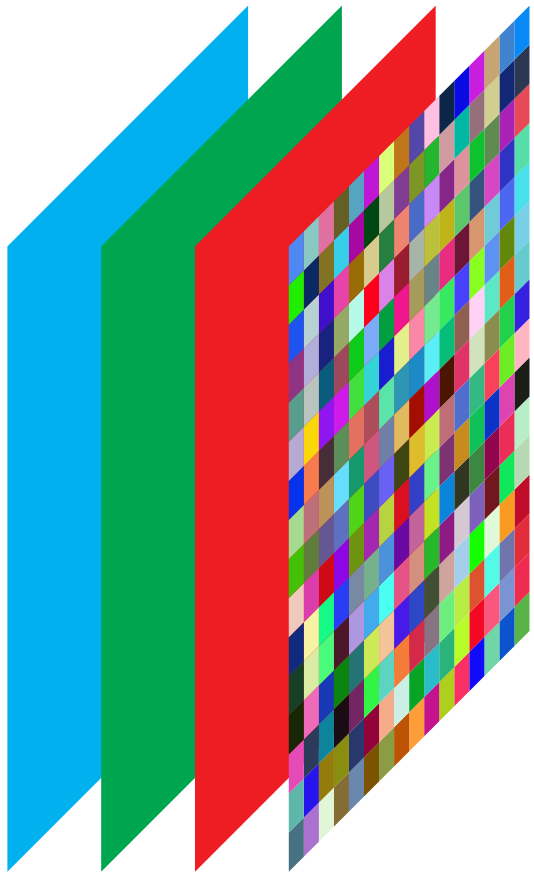
Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



Convolutional Neural Networks

- Convolutional Neural Networks (CNNs; ConvNets) are used in image recognition
- They assume their input are images



Convolutional Neural Networks

- We want to classify images into one of k classes

Convolutional Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features

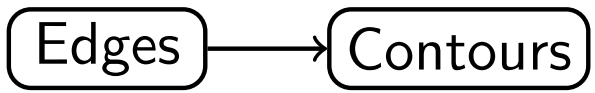
Convolutional Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features

Edges

Convolutional Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features

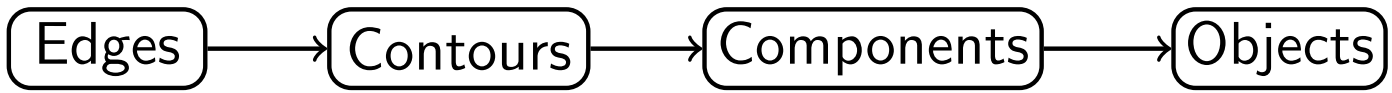


Convolutional Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features

Convolutional Neural Networks

- We want to classify images into one of k classes
- Extract hierarchical features



Convolutional Neural Networks

This is a cat ♥

Convolutional Neural Networks

Still a cat ♥♥

Convolutional Neural Networks

Half cat / half salad ♥♥♥

Convolutional Neural Networks

Minecraft cat ♥♥♥♥

Convolutional Neural Networks

- Our network learns to detect a feature in one part of the image
- Wouldn't it make sense to reuse that information?

Convolutional Neural Networks

- Our network learns to detect a feature in one part of the image
- Wouldn't it make sense to reuse that information?
- Yes!

Weight Sharing

Recipe for a Convolutional Neural Network

Recipe for a Convolutional Neural Network

- Ingredients

Recipe for a Convolutional Neural Network

- Ingredients
  1. Image I with dimension $w \times h \times d$

Recipe for a Convolutional Neural Network

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$

Recipe for a Convolutional Neural Network

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking

Recipe for a Convolutional Neural Network

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking
  - Put the image into the oven at 150°C

Recipe for a Convolutional Neural Network

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking
  - Don't put the image into the oven at 150°C

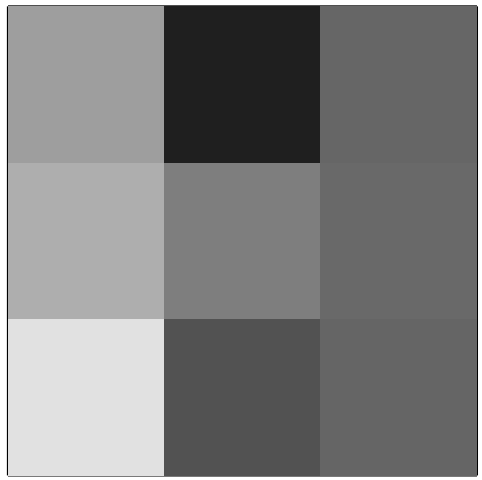
Recipe for a Convolutional Neural Network

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking
  - Don't put the image into the oven at 150°C
  - Slide the kernel across the image

Recipe for a Convolutional Neural Network

- Ingredients
  1. Image I with dimension $w \times h \times d$
  2. A kernel (filter) K of size $k \times l \times m$
- Cooking
  - Don't put the image into the oven at 150°C
  - Slide the kernel across the image
  - Compute the "dot product" for each configuration

Convolutional Neural Networks: Mechanics



Image

Convolutional Neural Networks: Mechanics

0.4	0.9	0.1
0.7	0.2	0.6
0.8	0.3	0.5

Image

Convolutional Neural Networks: Mechanics

0.4	0.9	0.1
0.7	0.2	0.6
0.8	0.3	0.5

Image

5.7	2.4
3.1	0.9

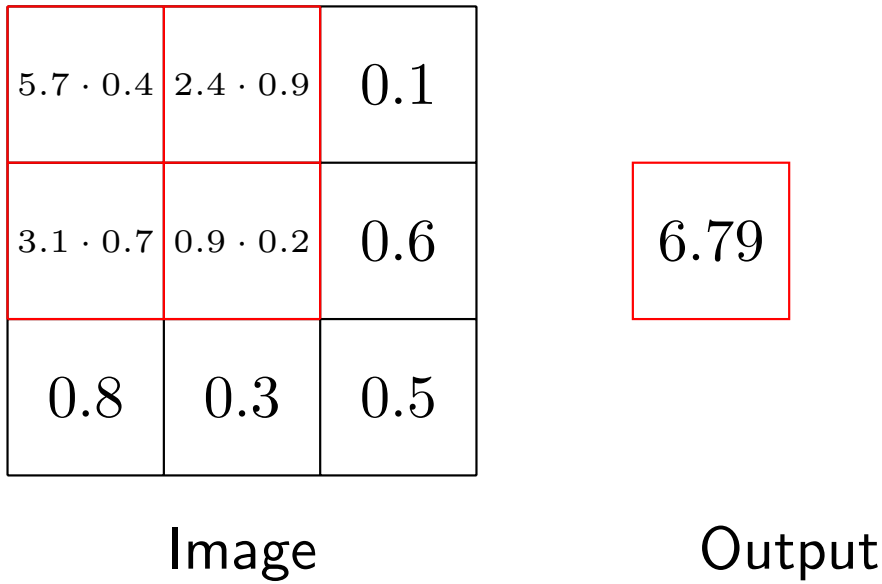
Kernel

Convolutional Neural Networks: Mechanics

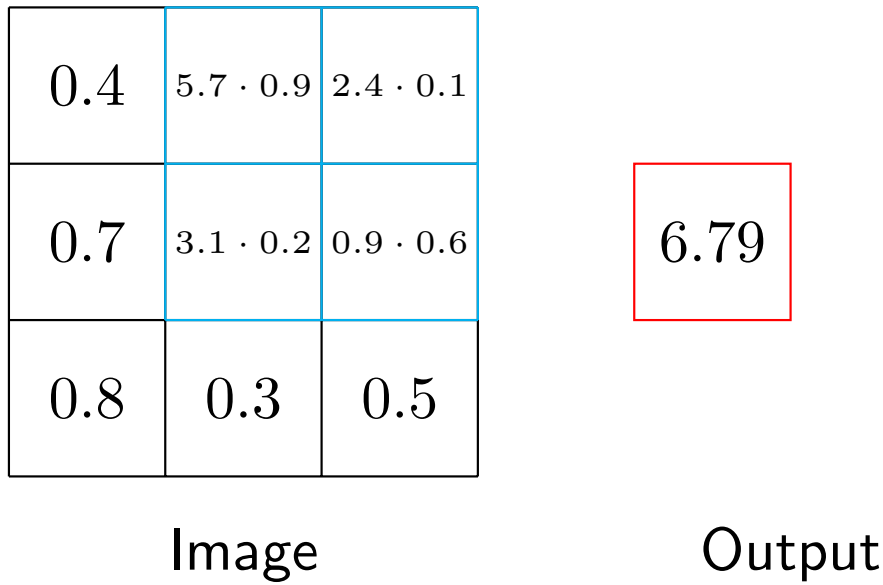
$5.7 \cdot 0.4$	$2.4 \cdot 0.9$	0.1
$3.1 \cdot 0.7$	$0.9 \cdot 0.2$	0.6
0.8	0.3	0.5

Image

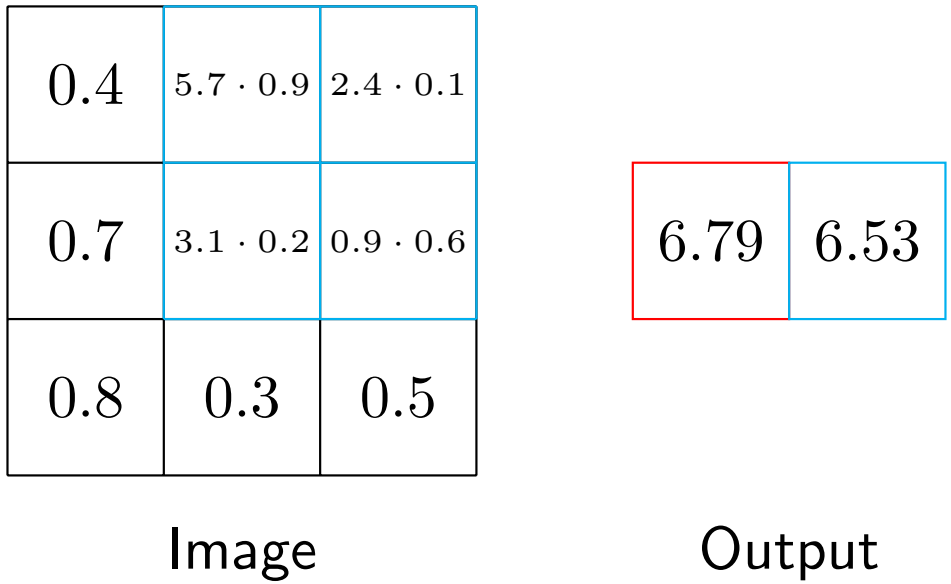
Convolutional Neural Networks: Mechanics



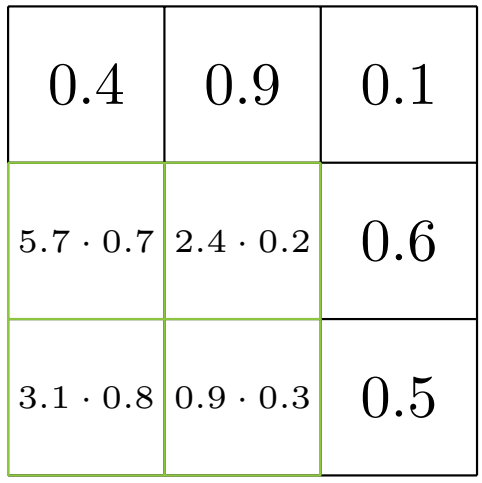
Convolutional Neural Networks: Mechanics



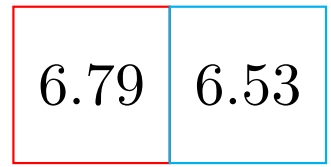
Convolutional Neural Networks: Mechanics



Convolutional Neural Networks: Mechanics

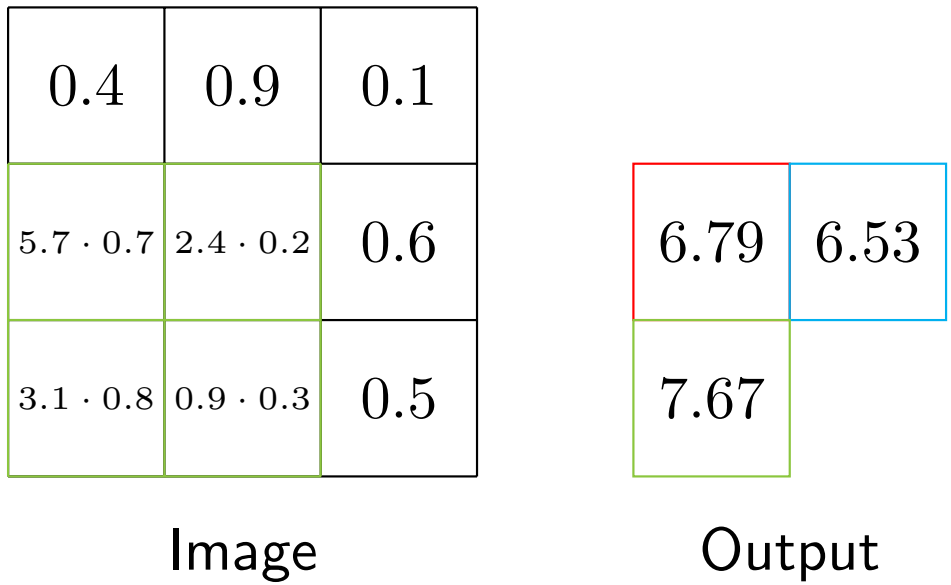


Image

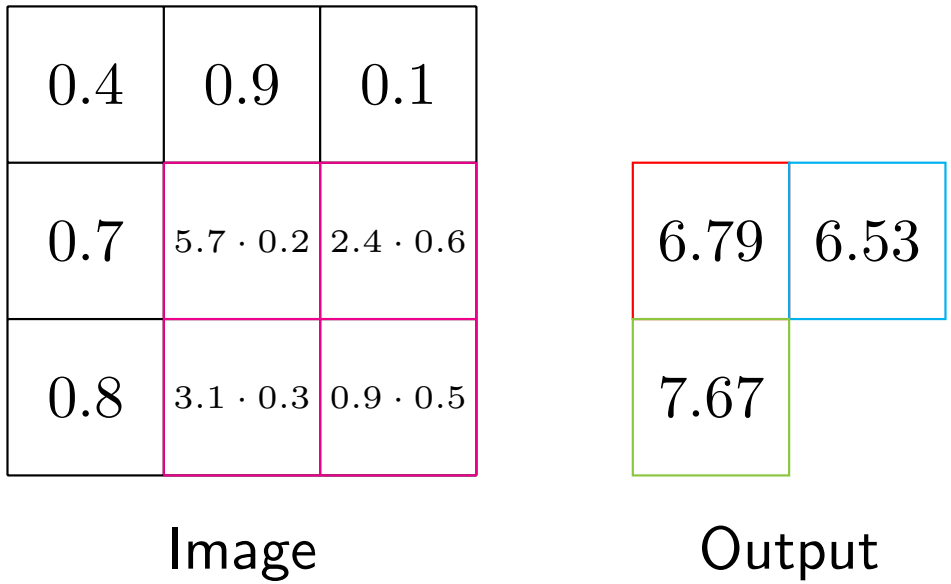


Output

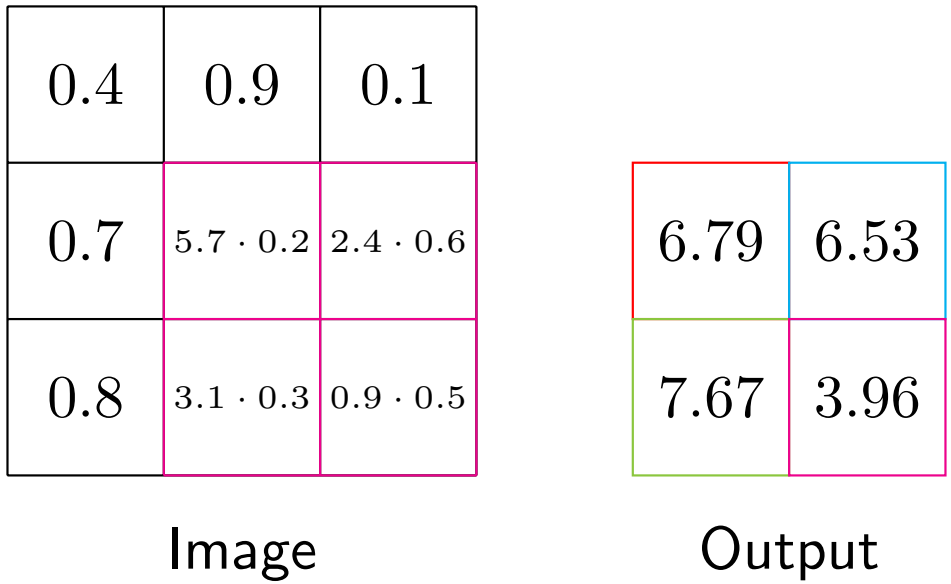
Convolutional Neural Networks: Mechanics



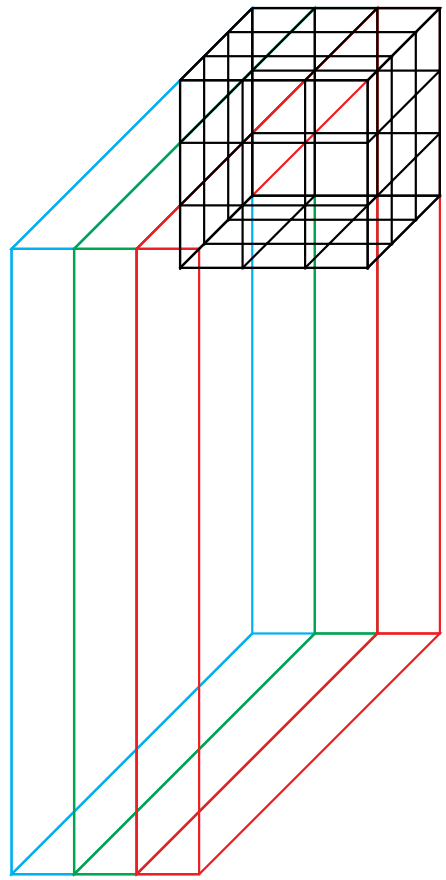
Convolutional Neural Networks: Mechanics



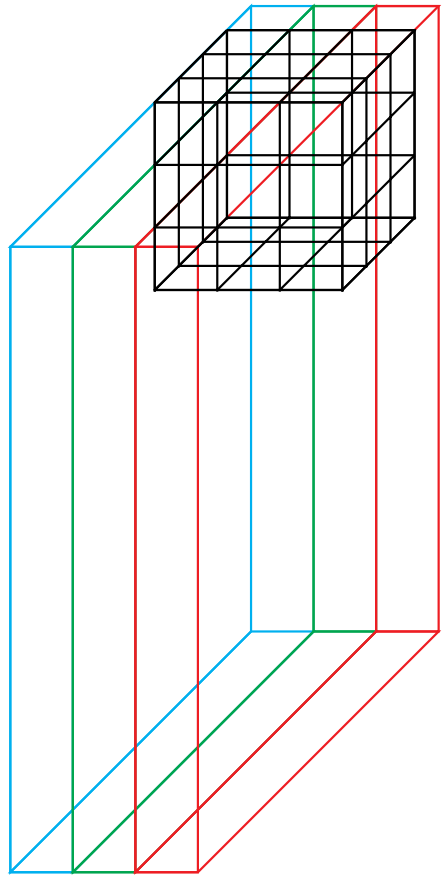
Convolutional Neural Networks: Mechanics



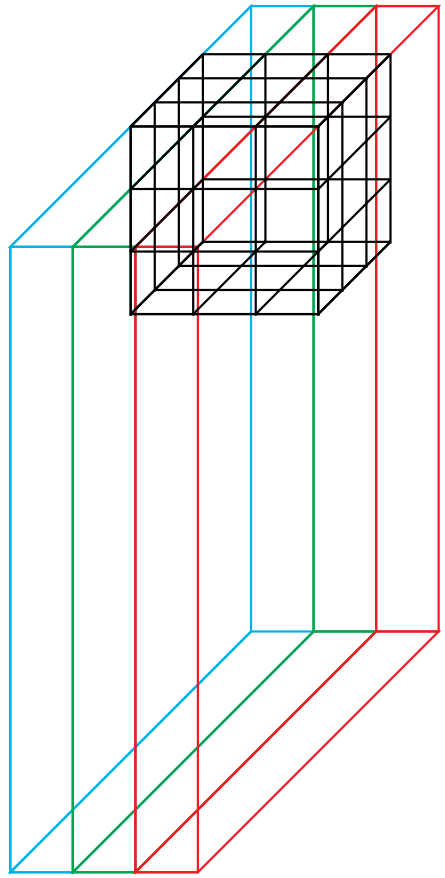
Convolutional Neural Networks: Mechanics



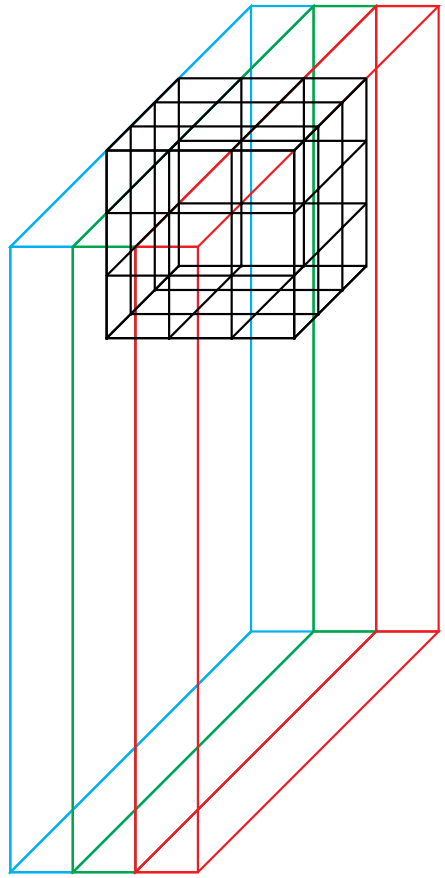
Convolutional Neural Networks: Mechanics



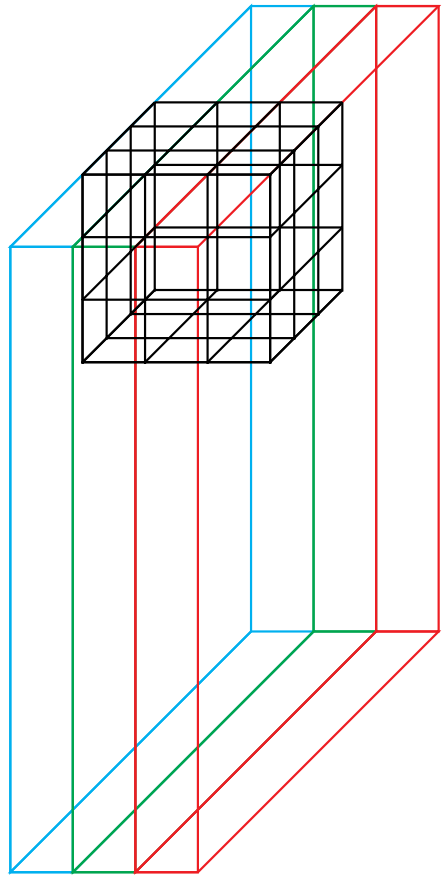
Convolutional Neural Networks: Mechanics



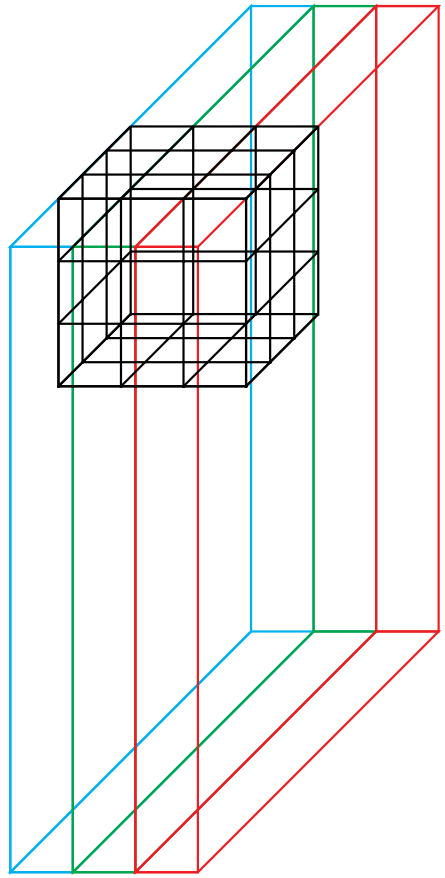
Convolutional Neural Networks: Mechanics



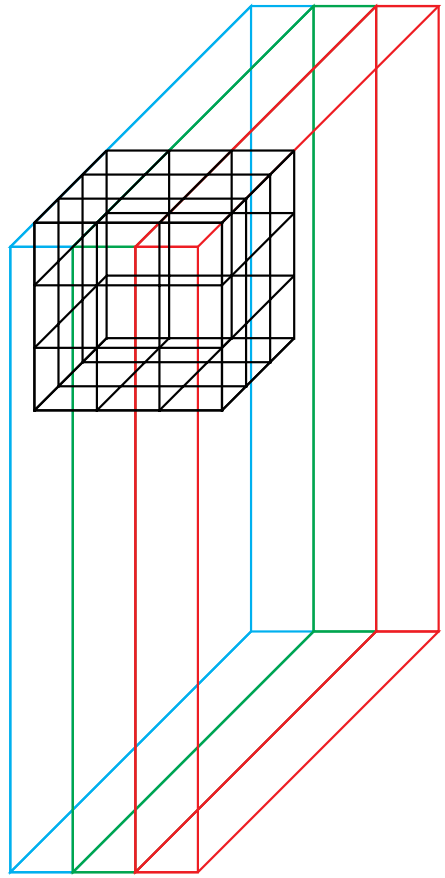
Convolutional Neural Networks: Mechanics



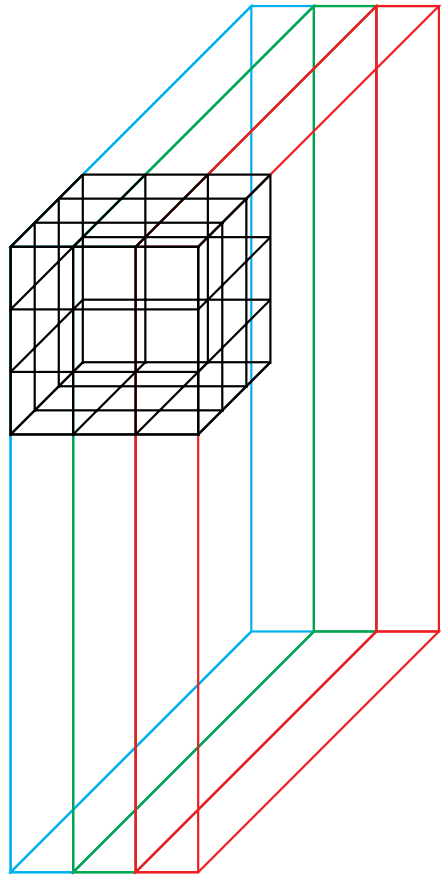
Convolutional Neural Networks: Mechanics



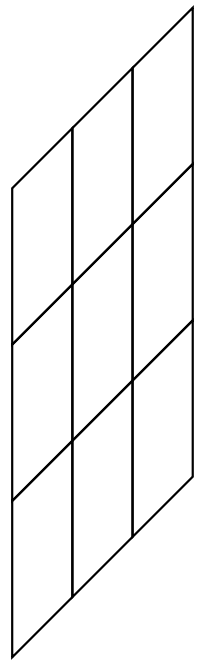
Convolutional Neural Networks: Mechanics



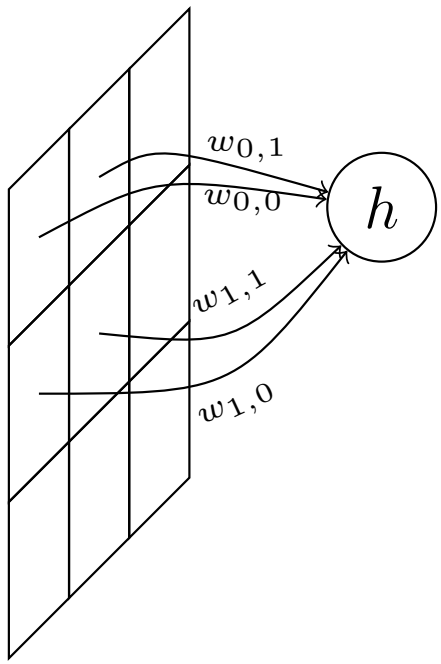
Convolutional Neural Networks: Mechanics



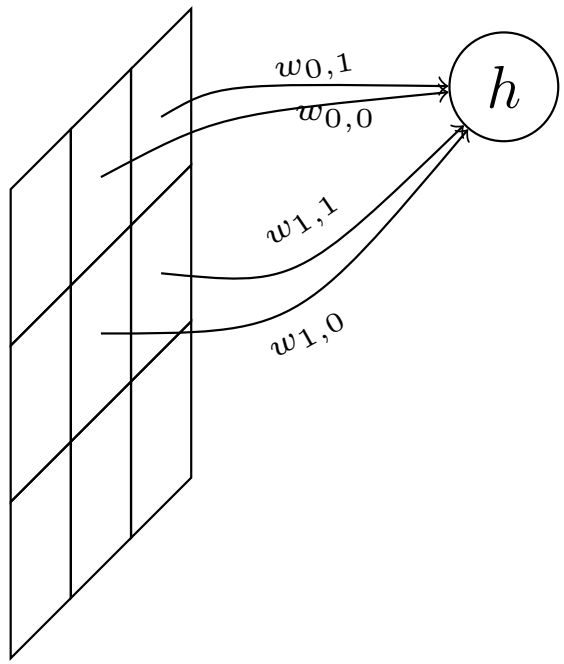
Convolutional Neural Networks: Mechanics



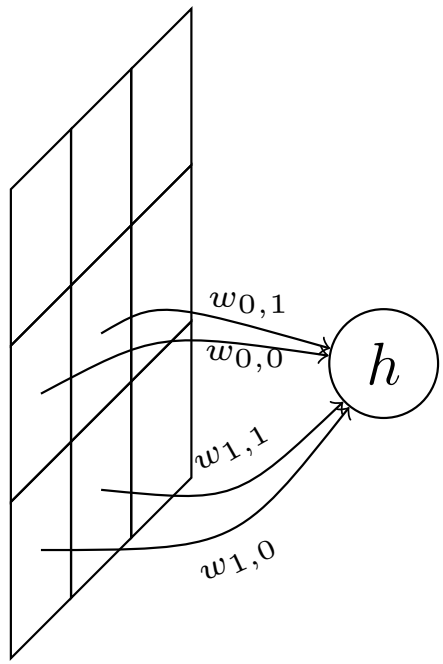
Convolutional Neural Networks: Mechanics



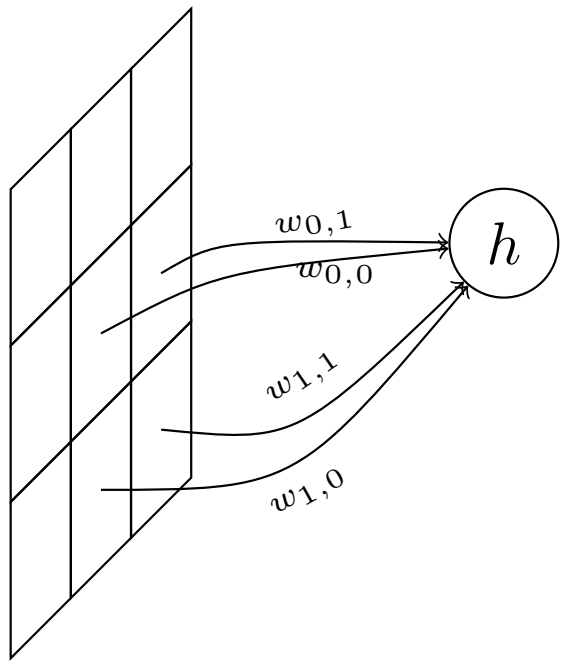
Convolutional Neural Networks: Mechanics



Convolutional Neural Networks: Mechanics



Convolutional Neural Networks: Mechanics



Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters

Convolutional Neural Networks: Hyperparameters

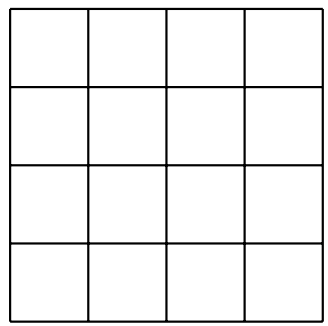
- Convolutional Neural Networks have three hyperparameters
  1. Kernel size

Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride

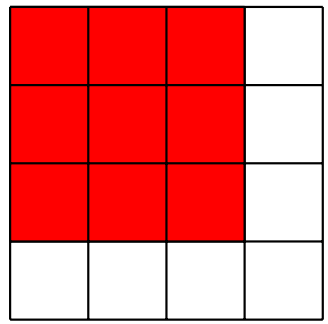
Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



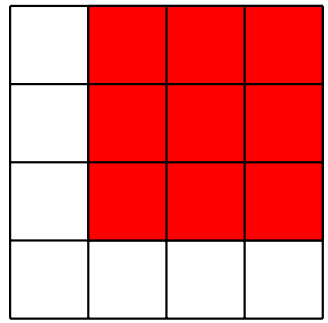
Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



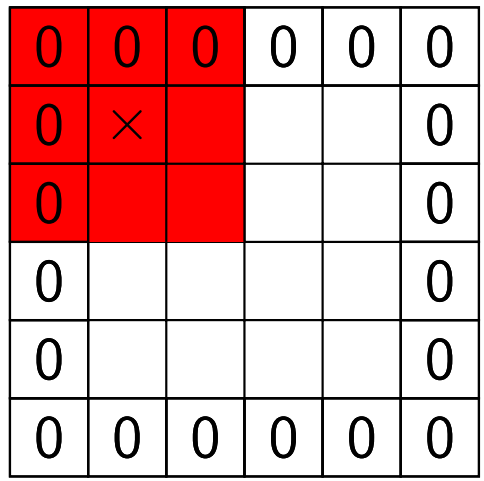
Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)

0	0	0	0	0	0
0					0
0					0
0					0
0					0
0	0	0	0	0	0

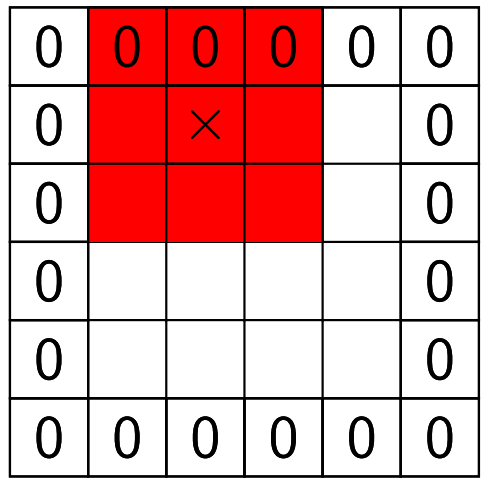
Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



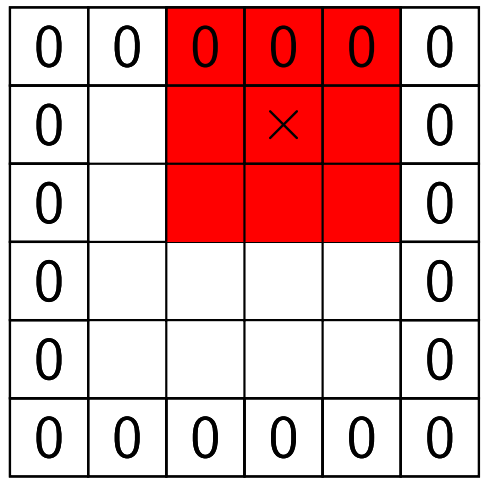
Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



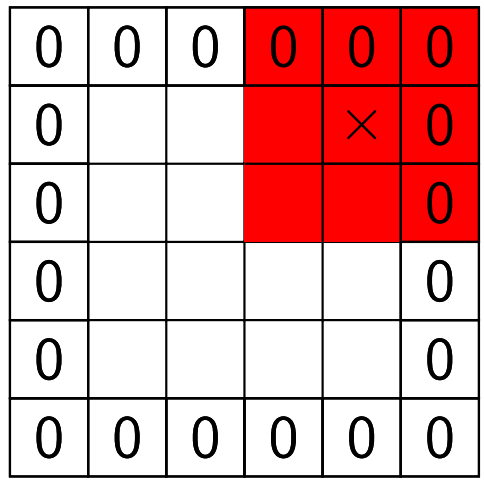
Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



Convolutional Neural Networks: Hyperparameters

- Convolutional Neural Networks have three hyperparameters
  1. Kernel size
  2. Kernel stride
  3. Padding (valid or same)



Convolutional Neural Networks: Intuition

Convolutional Neural Networks: Intuition

- Each layer of a ConvNet learns to detect features

Convolutional Neural Networks: Intuition

- Each layer of a ConvNet learns to detect features
- Later layers combine features of earlier layers

Convolutional Neural Networks: Intuition

What's in a kernel?

0	1	0
0	1	0
0	1	0

Patterns

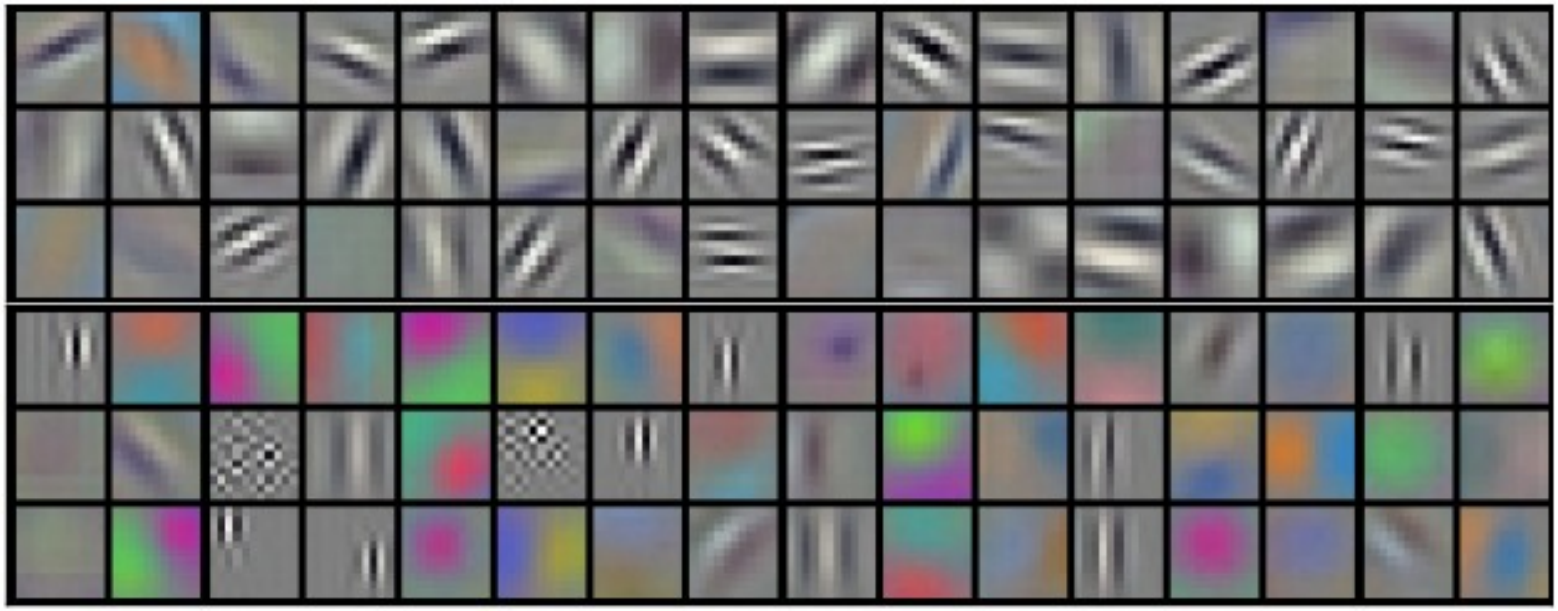
Convolutional Neural Networks: Intuition

What's in a kernel?

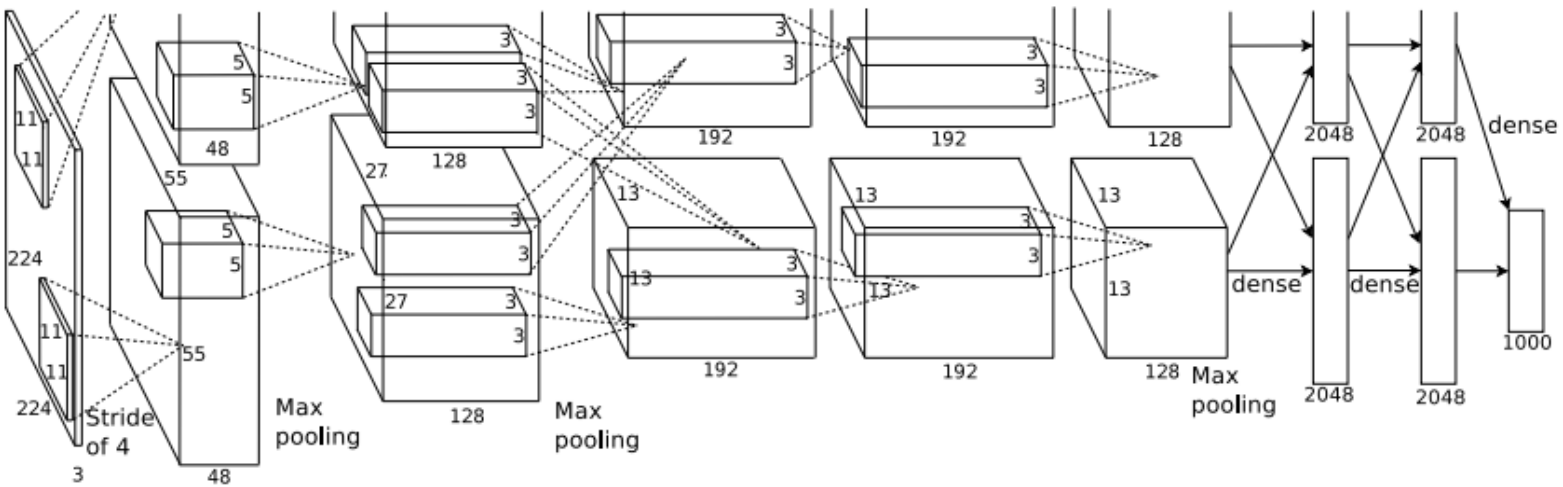
0	0	0
-1	1	0
0	0	0

Features

Convolutional Neural Networks: Intuition



Case Study: AlexNet



Sequences

Sequences

- Humans think in sequences

Sequences

- Humans think in sequences
- Sequences give single entities context

I eat people

Sequences

- Humans think in sequences
- Sequences give single entities context

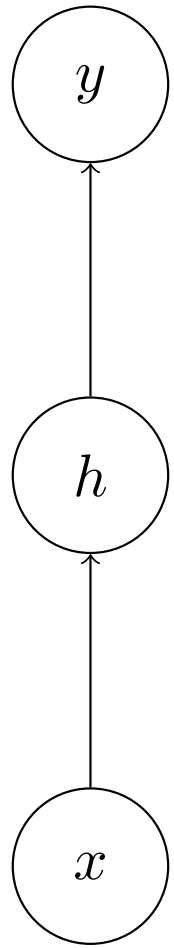
I enjoy eating dinner with people

Sequences

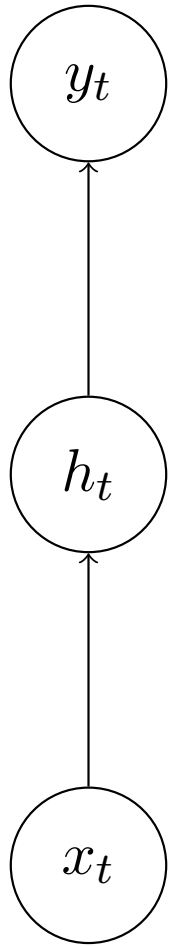
- Humans think in sequences
- Sequences give single entities context
- The key to understanding sequences is memory

I enjoy eating dinner with people

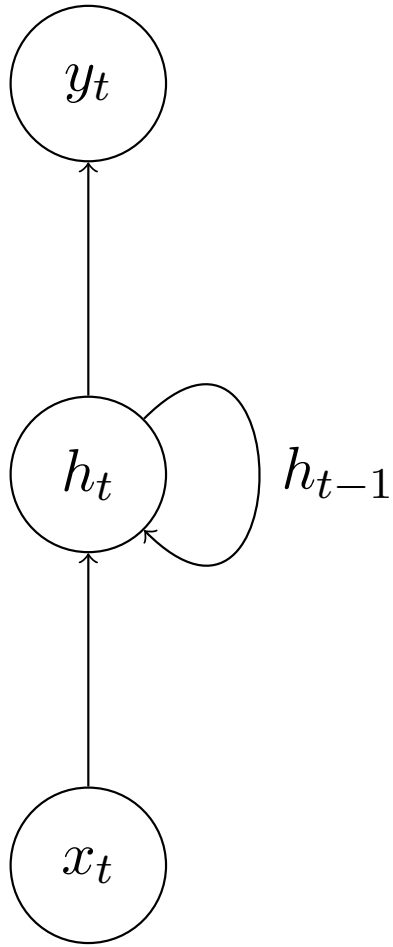
Machine Memory



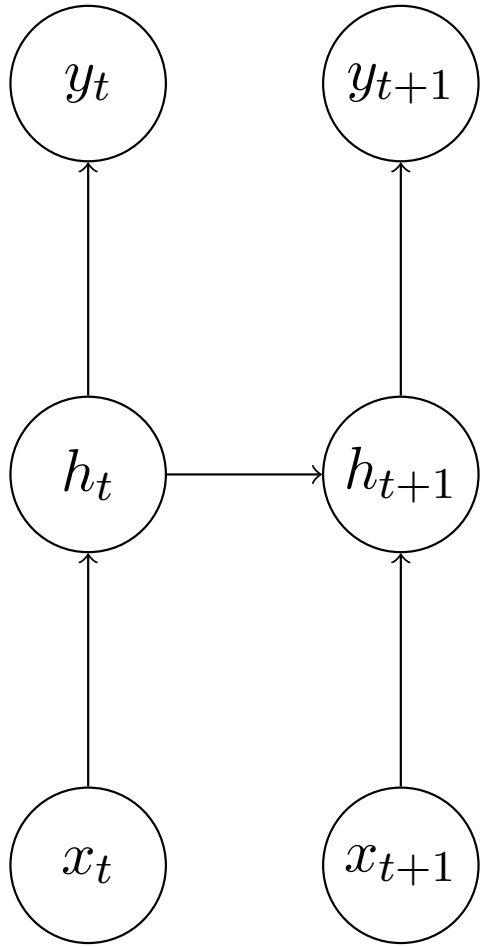
Machine Memory



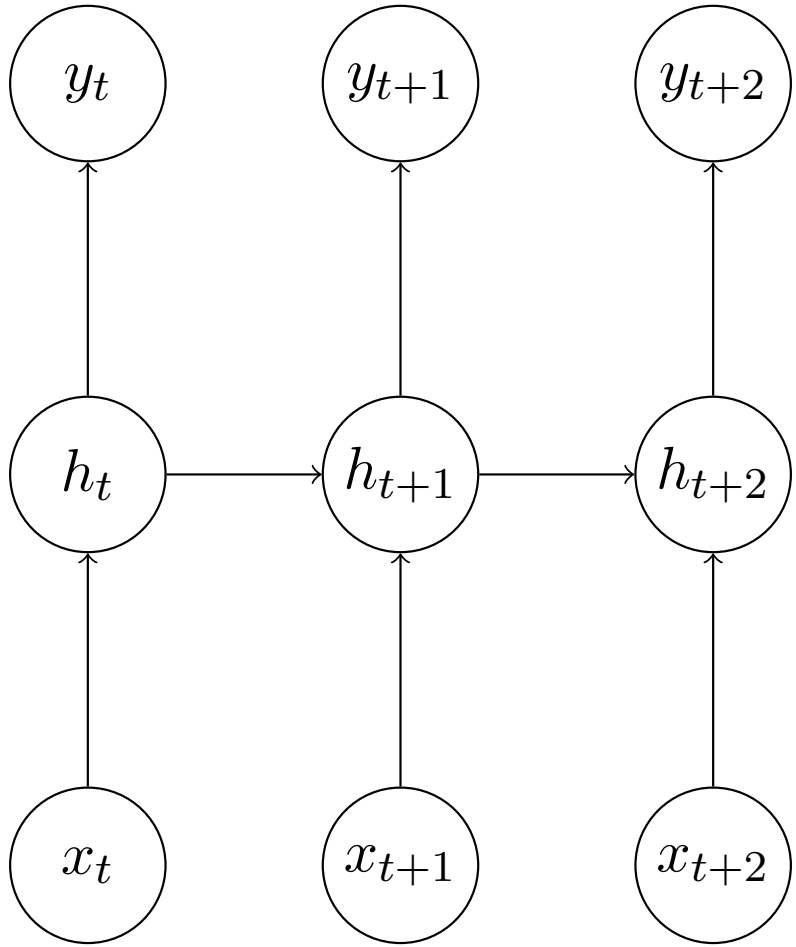
Machine Memory



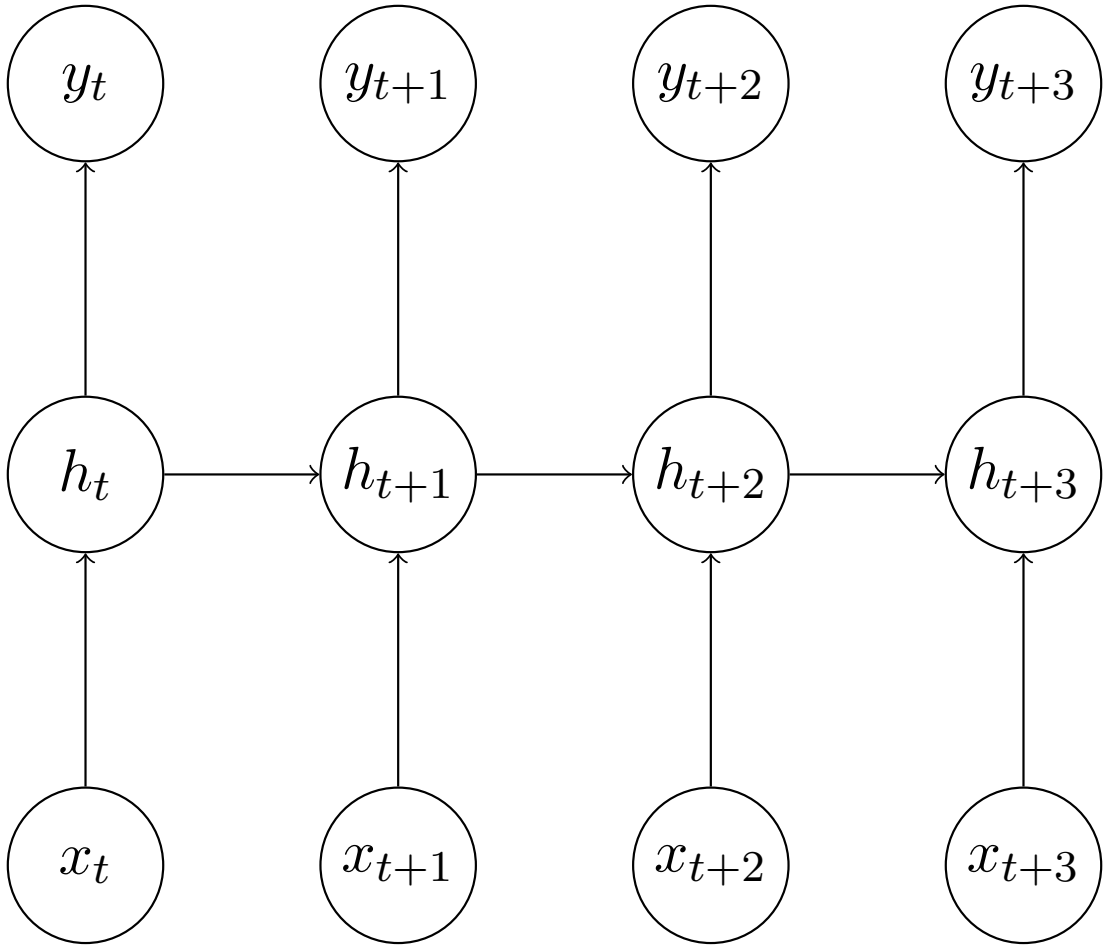
Machine Memory



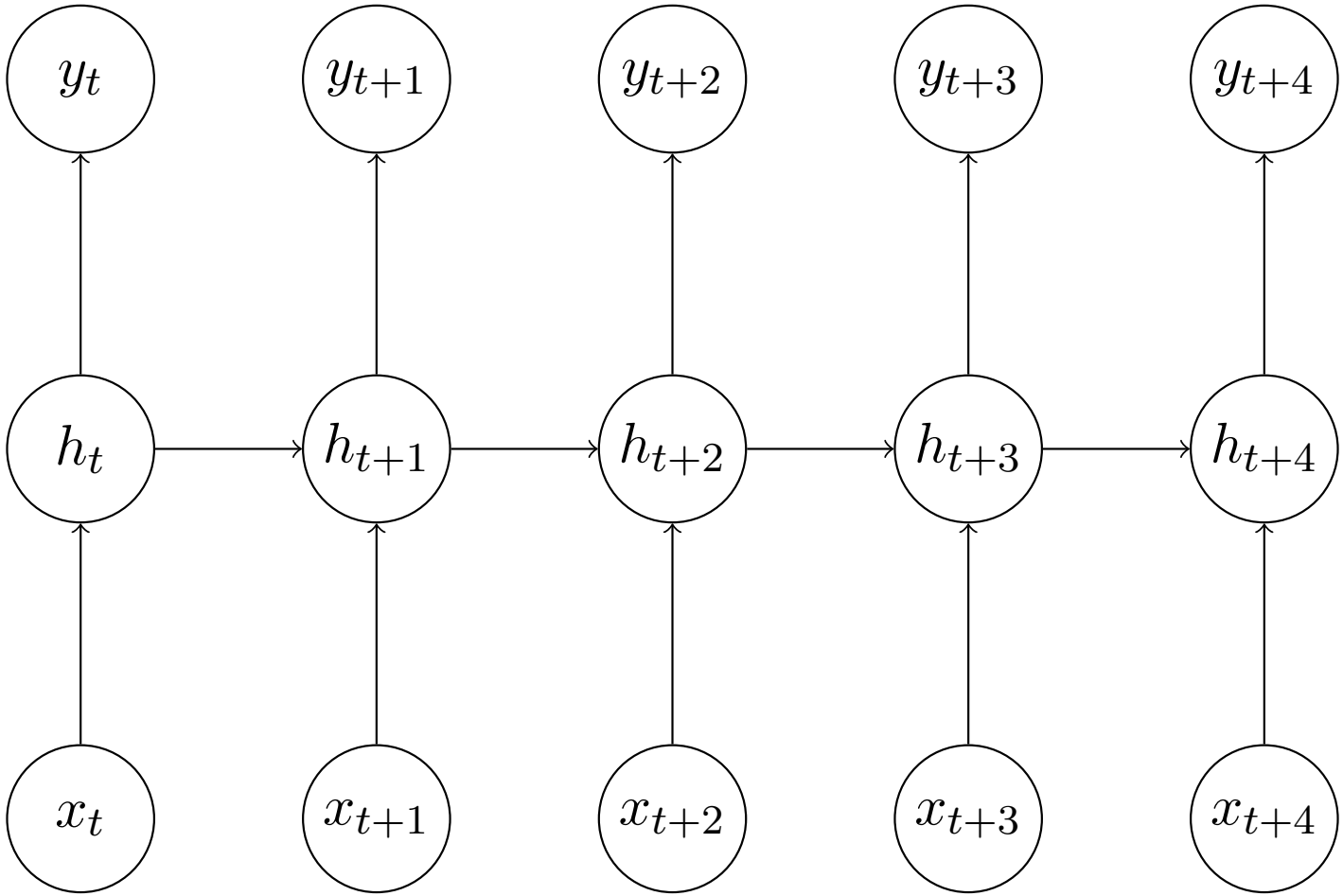
Machine Memory



Machine Memory



Machine Memory



Machine Memory

$$h_t = f(w \cdot x + b)$$

Machine Memory

$$h_t = f(\mathbf{w}^\top [h_{t-1}, x] + b)$$

Recurrent Neural Networks

Recurrent Neural Networks

- Recurrent Neural Networks (RNNs) share weights through time

Recurrent Neural Networks

- Recurrent Neural Networks (RNNs) share weights through time
- They have *memory*

Recurrent Neural Networks

- Recurrent Neural Networks (RNNs) share weights through time
- They have *memory*
- And they have a problem:

The Vanishing Gradient Problem

LSTMs

- RNN's are forgetful

LSTMs

- RNN's are forgetful
- *Long Short Term Memory* (LSTM) Units solve this problem

LSTMs

- RNN's are forgetful
- *Long Short Term Memory* (LSTM) Units solve this problem
- Developed by Schmidhuber and Hochreiter at TUM in 1997

LSTMs

LSTMs

- LSTMs are a lot like flip-flops

LSTMs

- LSTMs are a lot like flip-flops
- They have three *gates*

LSTMs

- LSTMs are a lot like flip-flops
- They have three *gates*
  - Input gate

LSTMs

- LSTMs are a lot like flip-flops
- They have three *gates*
  - Input gate
  - Forget gate

LSTMs

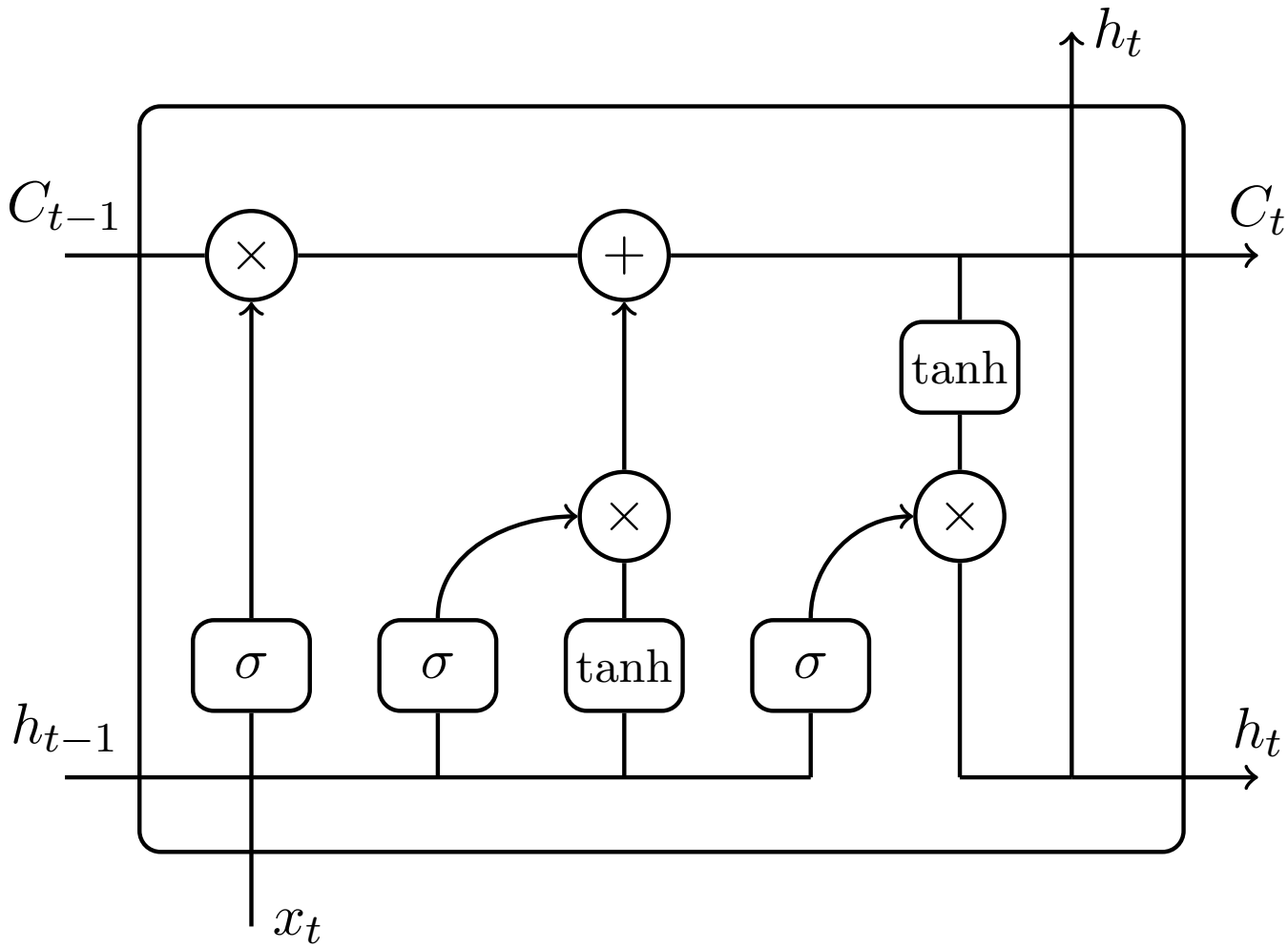
- LSTMs are a lot like flip-flops
- They have three *gates*
  - Input gate
  - Forget gate
  - Output gate

LSTMs

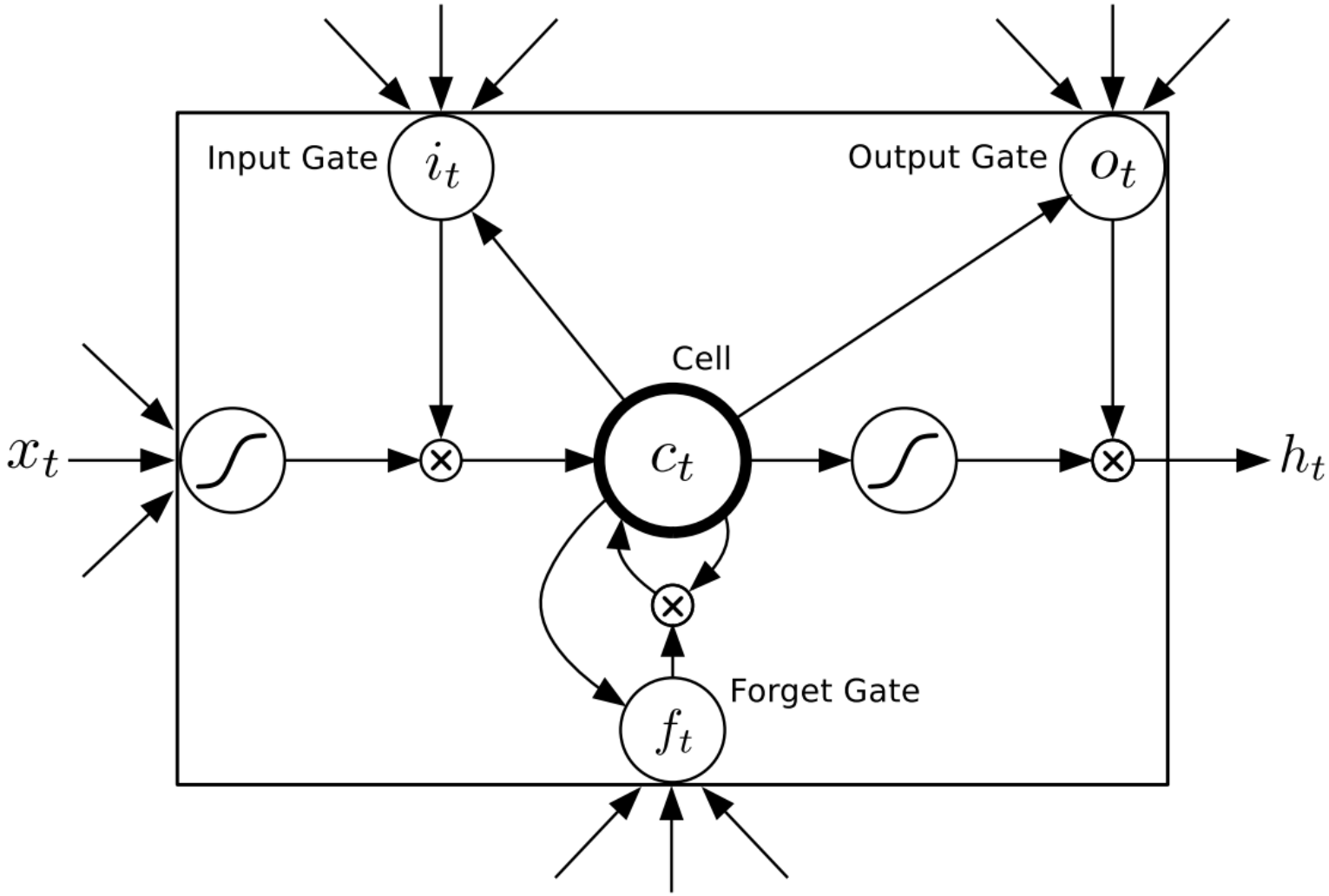
- LSTMs are a lot like flip-flops
- They have three *gates*
  - Input gate
  - Forget gate
  - Output gate

$$G_i(x_t, h_{t-1}) = \sigma(\mathbf{w}_i^\top [x_t, h_{t-1}] + b_i)$$

LSTMs



LSTMs



;

LSTMs: What can they do?

So, what can LSTMs actually do?

LSTMs: What can they do?

tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e plia tklrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

Iteration: 100

[Kar16a]

LSTMs: What can they do?

"Tmont thithey" fomesscerliund Keushey. Thom here sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

Iteration: 300

[Kar16a]

LSTMs: What can they do?

we counter. He stutn co des. His stanted out one ofler that concossions and was to gearang reay Jotrets and with fre colt otf paitt thin wall. Which das stimn

Iteration: 500

[Kar16a]

LSTMs: What can they do?

Aftair fall unsuch that the hall for Prince Velzonski's that me of her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort how, and Gogition is so overelical and ofter.

Iteration: 700

[Kar16a]

LSTMs: What can they do?

"Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had oftened him. Pierre aking his soul came to the packs and drove up his father-in-law women.

Iteration: 2000

[Kar16a]

LSTMs: What can they do?

They can write Linux kernel code!

LSTMs: What can they do?

They can compose music!

Deep Learning

- Why the recent success of deep learning?

Deep Learning

- Why the recent success of deep learning?
- Three reasons

Deep Learning

- Why the recent success of deep learning?
- Three reasons
  1. Better hardware

Deep Learning

- Why the recent success of deep learning?
- Three reasons
  1. Better hardware
  2. More data

Deep Learning

- Why the recent success of deep learning?
- Three reasons
  1. Better hardware
  2. More data
  3. Better methods

The Ugly

TensorFlow

Feel the TensorFlow

A Tour of TensorFlow

A Tour of TensorFlow

TensorFlow is

A Tour of TensorFlow

TensorFlow is

- An open source deep learning library

A Tour of TensorFlow

TensorFlow is

- An open source deep learning library
- Released by Google in November 2015

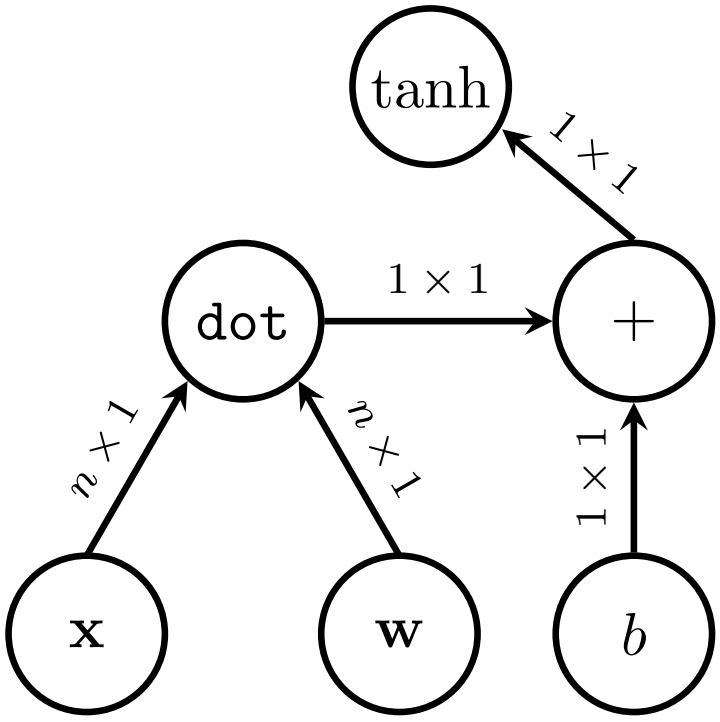
A Tour of TensorFlow

TensorFlow is

- An open source deep learning library
- Released by Google in November 2015
- Especially suited to:
  - "Large-scale machine learning on
  - heterogeneous distributed systems"

Computational Paradigms

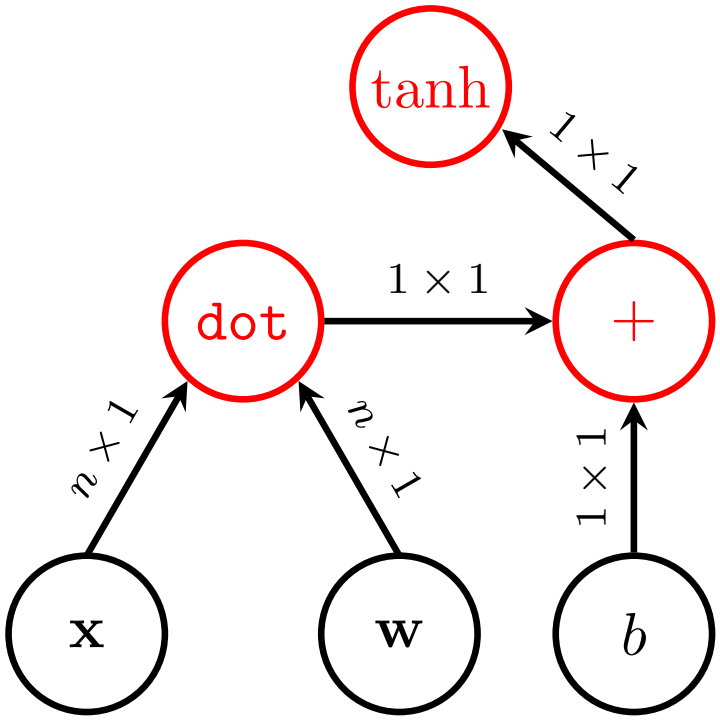
Computational Paradigms



$$\hat{y} = \tanh(\mathbf{x}^\top \mathbf{w} + b)$$

Computational Graphs

Computational Paradigms

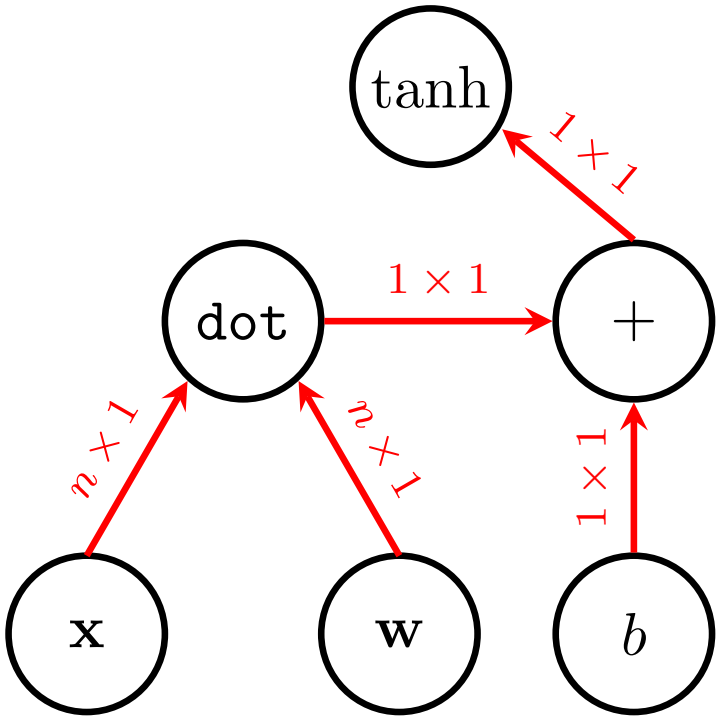


$$\hat{y} = \tanh(\mathbf{x}^\top \mathbf{w} + b)$$

Computational Graphs

1. Operations

Computational Paradigms

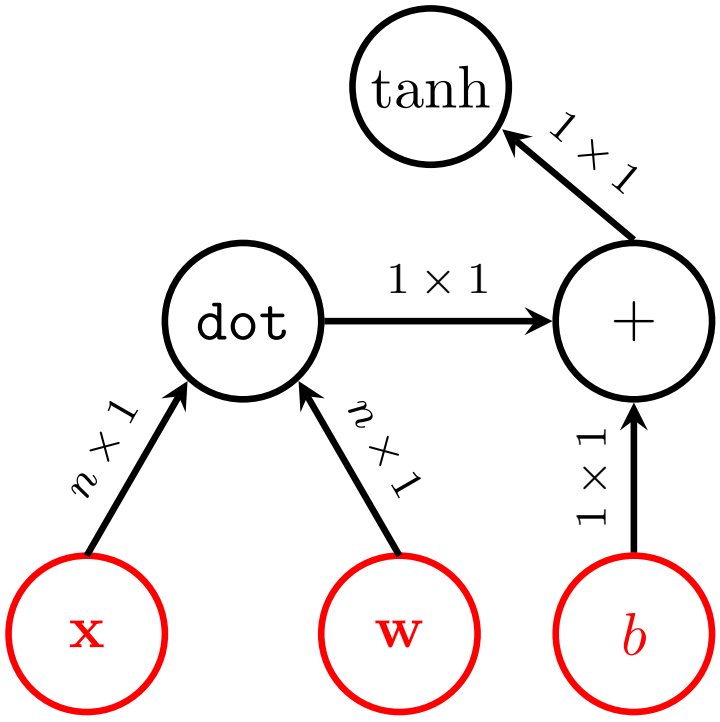


$$\hat{y} = \tanh(\mathbf{x}^\top \mathbf{w} + b)$$

Computational Graphs

1. Operations
2. Tensors

Computational Paradigms

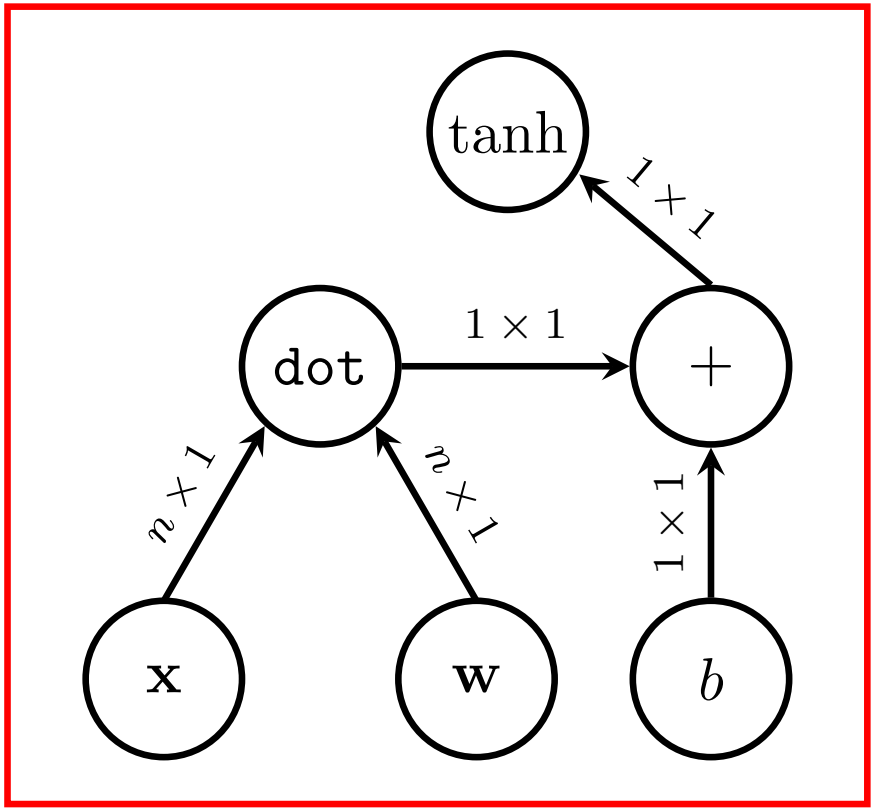


$$\hat{y} = \tanh(\mathbf{x}^\top \mathbf{w} + b)$$

Computational Graphs

1. Operations
2. Tensors
3. Variables

Computational Paradigms



$$\hat{y} = \texttt{session.run}(\tanh(\mathbf{x}^\top \mathbf{w} + b))$$

Computational Graphs

1. Operations
2. Tensors
3. Variables
4. Sessions

Execution Model

Execution Model

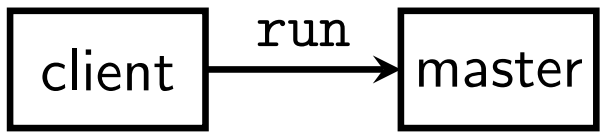
Actors

Execution Model

Actors

1. Client

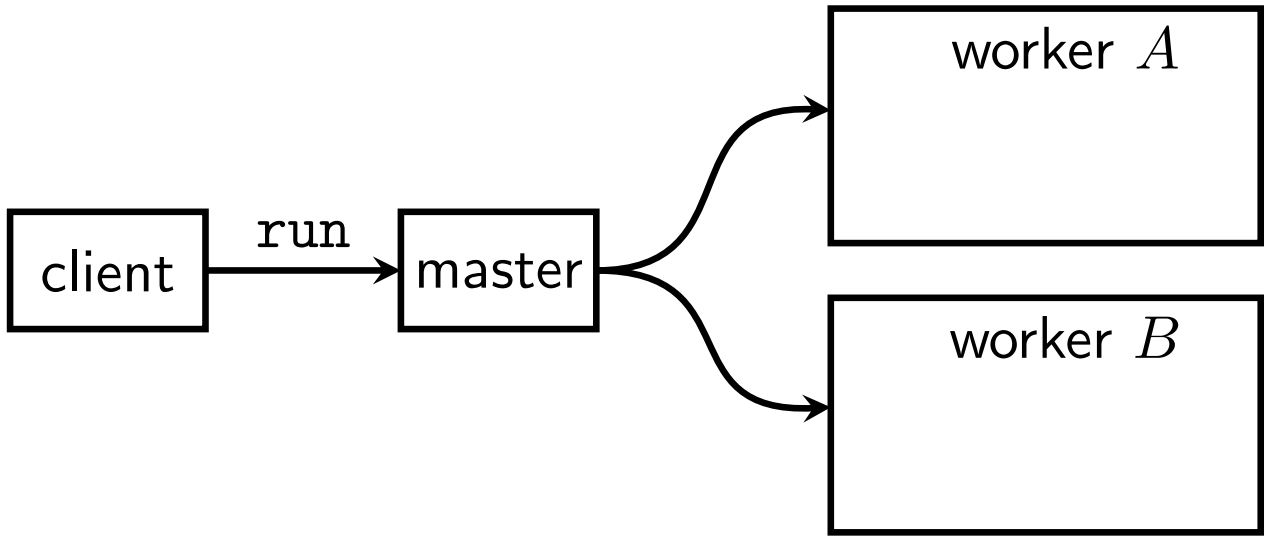
Execution Model



Actors

1. Client
2. Master

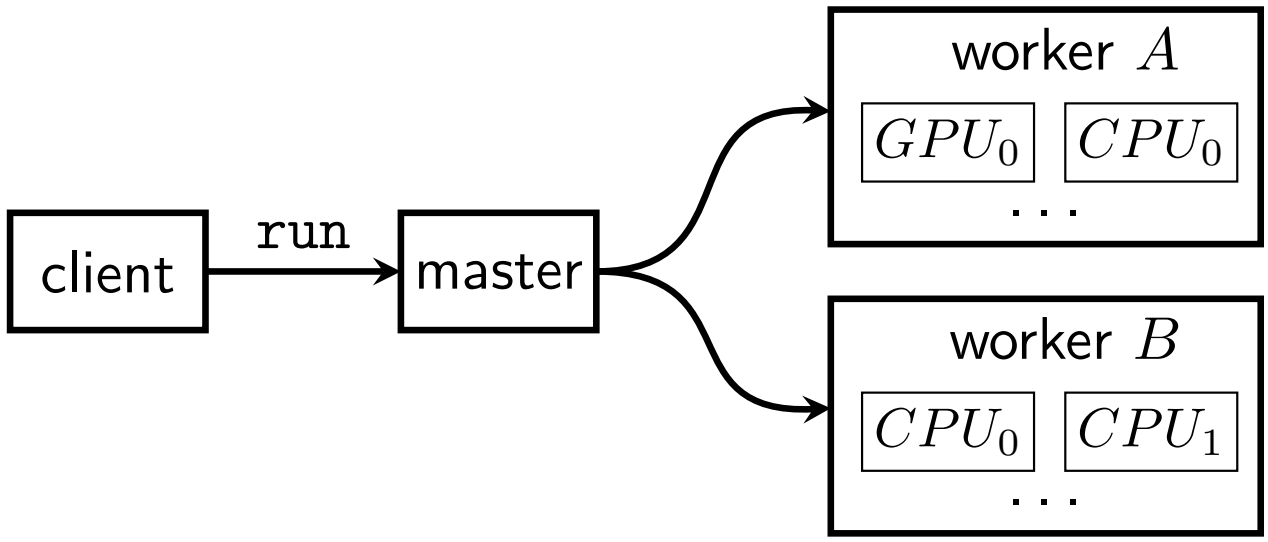
Execution Model



Actors

1. Client
2. Master
3. Workers

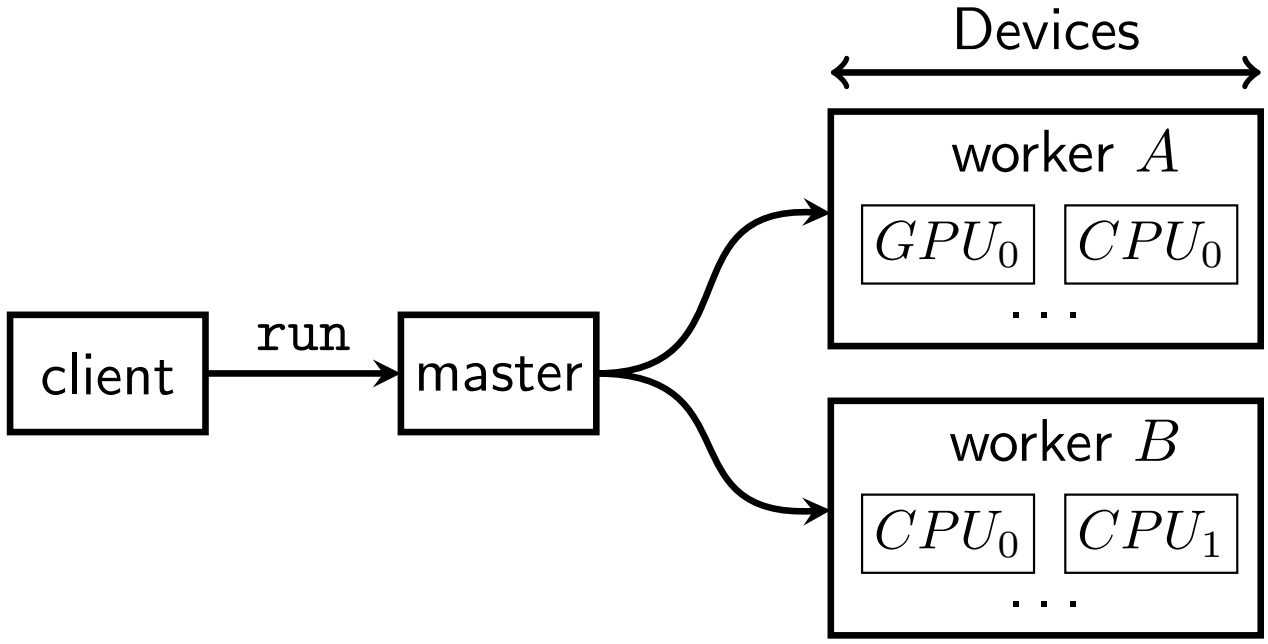
Execution Model



Actors

1. Client
2. Master
3. Workers
4. Devices

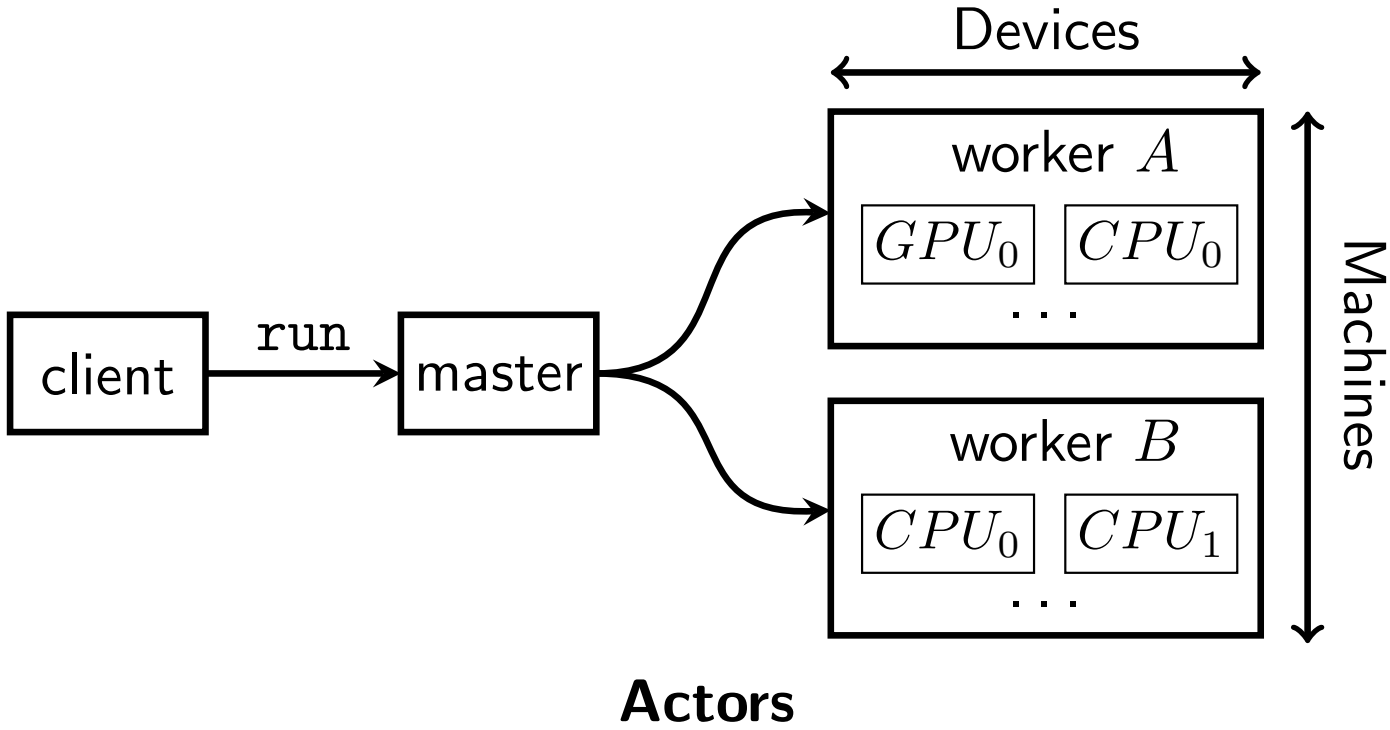
Execution Model



Actors

1. Client
2. Master
3. Workers
4. Devices

Execution Model



1. Client
2. Master
3. Workers
4. Devices

Visualization Tools

Visualization Tools

- Deep Neural Networks have the tendency of being . . . deep

Visualization Tools

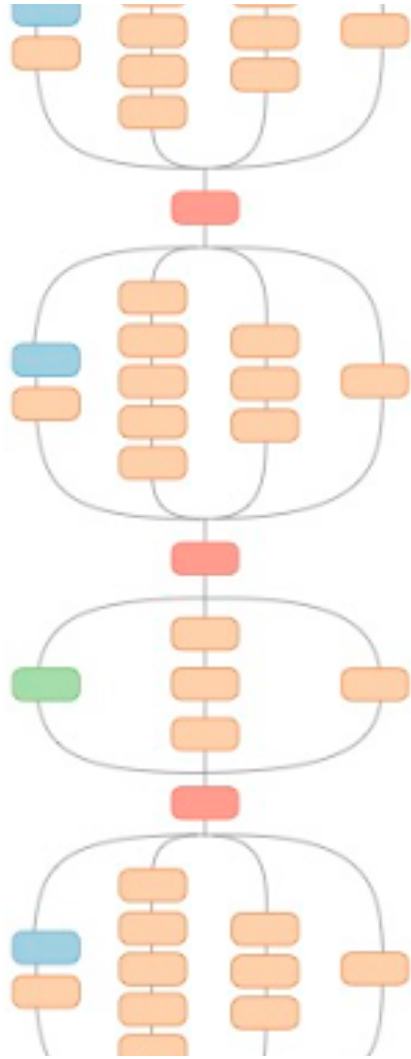
- Deep Neural Networks have the tendency of being . . . deep
- Easy to drown in the complexity of an architecture

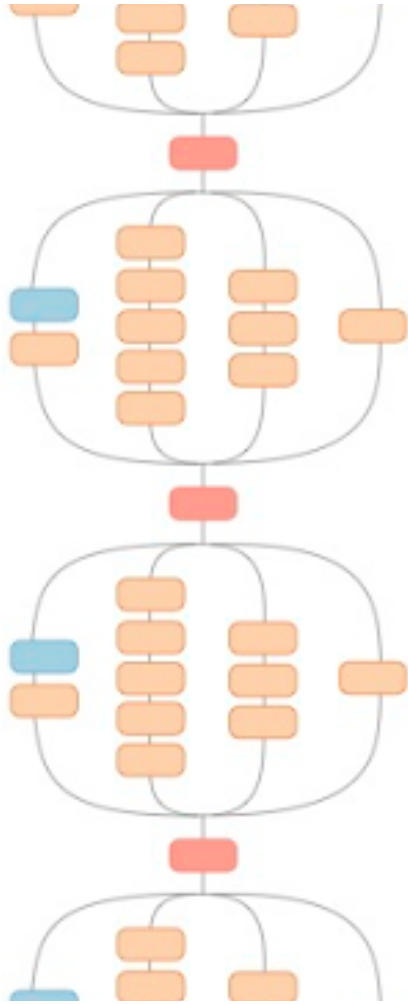
Visualization Tools

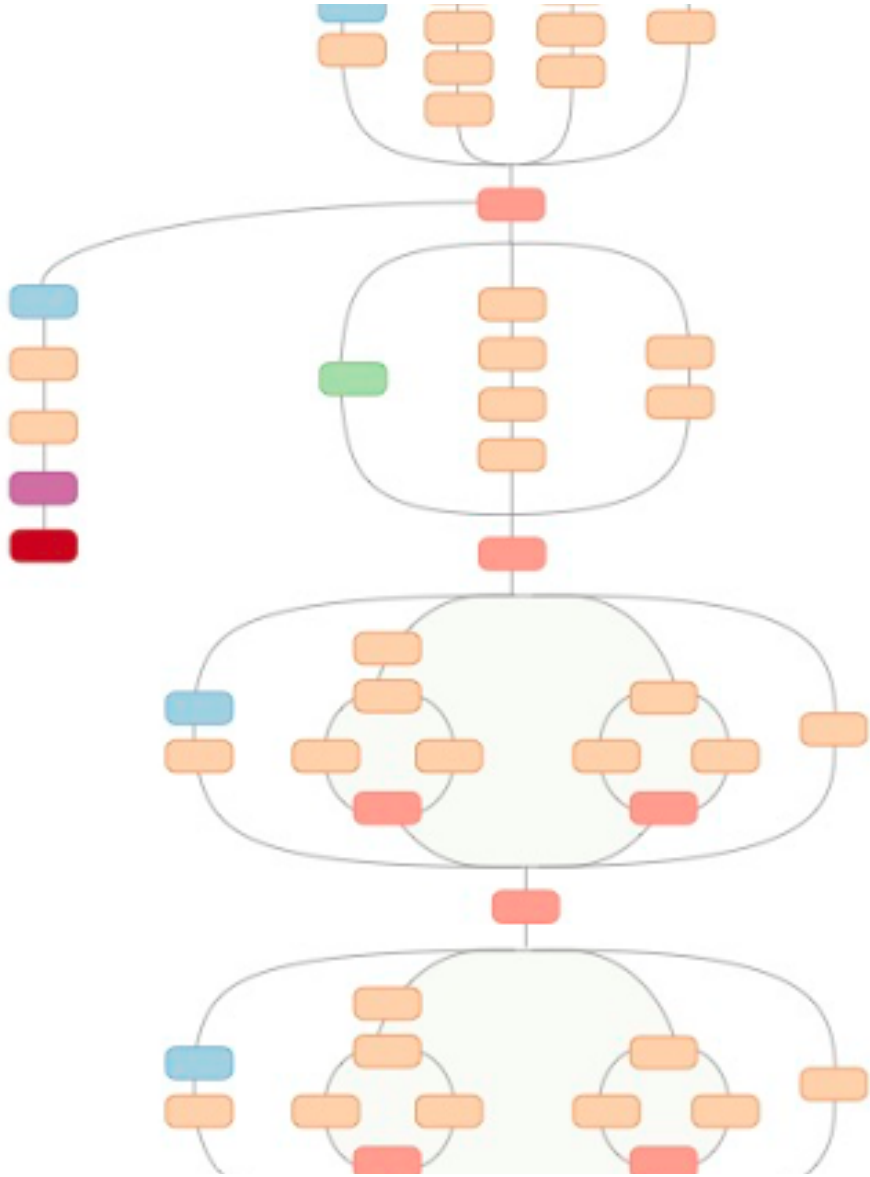
- Deep Neural Networks have the tendency of being . . . deep
- Easy to drown in the complexity of an architecture
- $>$ 36,000 nodes for Google's *Inception* model

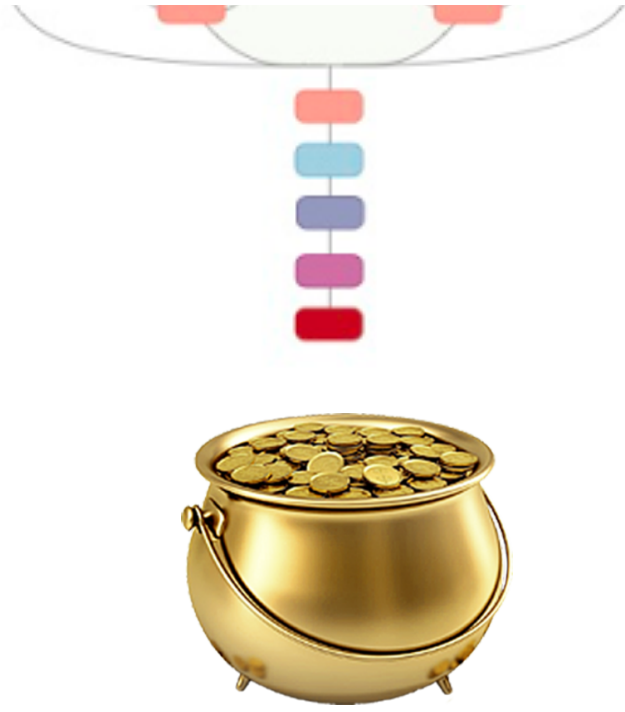
Visualization Tools

- Deep Neural Networks have the tendency of being . . . deep
- Easy to drown in the complexity of an architecture
- $>$ 36,000 nodes for Google's *Inception* model









Source: http://googleresearch.blogspot.de/2016/03/train-your-own-image-classifier-with.html

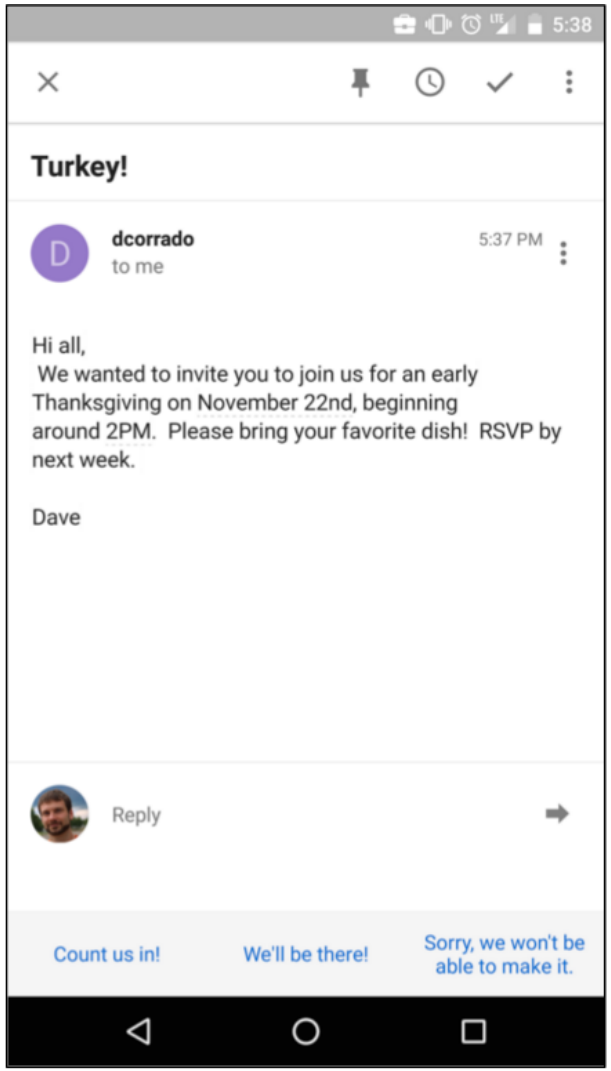
TensorBoard to the Rescue

Use Cases

Source: http://googleresearch.blogspot.de/2015/11/computer-respond-to-this-email.html

Use Cases

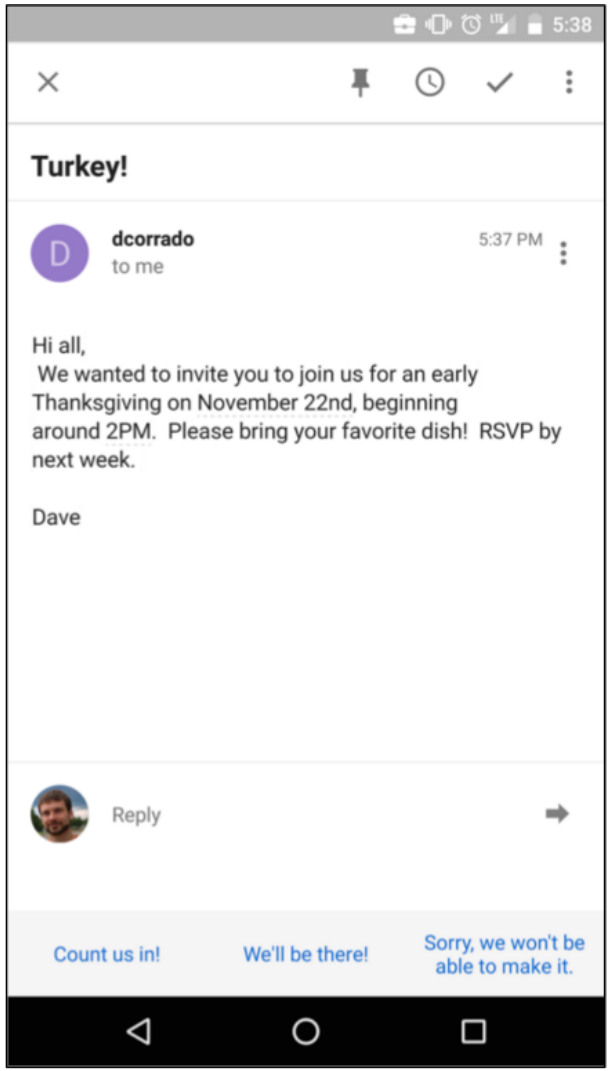
- Smart email replies in Google *Inbox*



Source: http://googleresearch.blogspot.de/2015/11/computer-respond-to-this-email.html

Use Cases

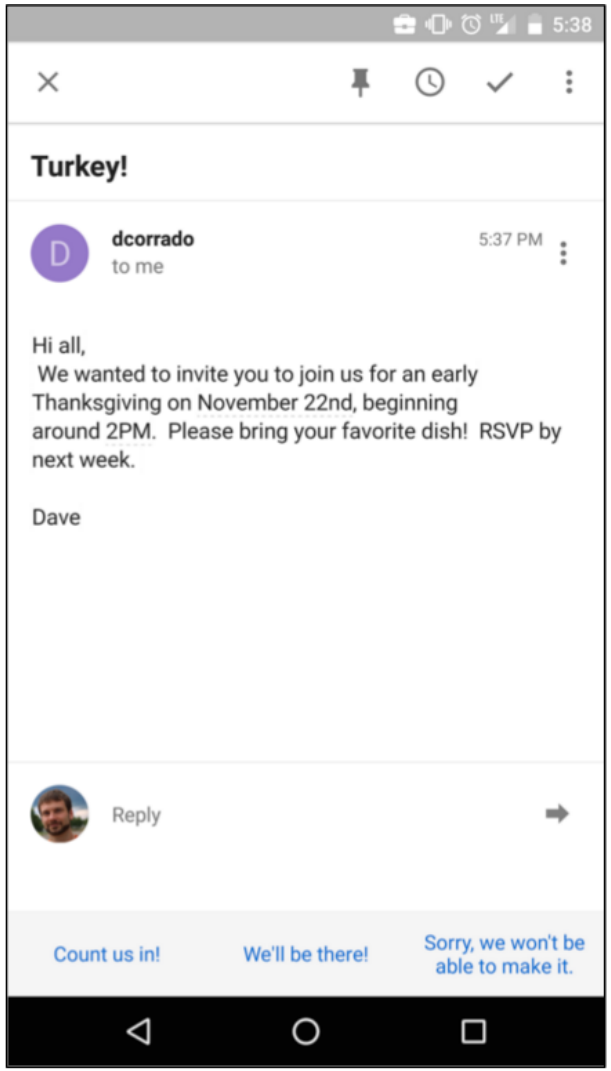
- Smart email replies in Google *Inbox*
- Emails mapped to "thought vectors"



Source: http://googleresearch.blogspot.de/2015/11/computer-respond-to-this-email.html

Use Cases

- Smart email replies in Google *Inbox*
- Emails mapped to "thought vectors"
- LSTMs synthesize valid replies



Source: http://googleresearch.blogspot.de/2015/11/computer-respond-to-this-email.html

Use Cases

Use Cases

- Google DeepMind now using TensorFlow

Use Cases

- Google DeepMind now using TensorFlow
- Already for *AlphaGo*

Source: https://deepmind.com/css/images/opengraph/alphago-logo.png

Use Cases

- Google DeepMind now using TensorFlow
- Already for *AlphaGo*
- According to a DeepMind SWE reasons are:

Source: https://deepmind.com/css/images/opengraph/alphago-logo.png

Use Cases

- Google DeepMind now using TensorFlow
- Already for *AlphaGo*
- According to a DeepMind SWE reasons are:
  - Python,

Source: https://deepmind.com/css/images/opengraph/alphago-logo.png

Use Cases

- Google DeepMind now using TensorFlow
- Already for *AlphaGo*
- According to a DeepMind SWE reasons are:
  - Python,
  - Integration with Google Cloud Platform,

Source: https://deepmind.com/css/images/opengraph/alphago-logo.png

Use Cases

- Google DeepMind now using TensorFlow
- Already for *AlphaGo*
- According to a DeepMind SWE reasons are:
  - Python,
  - Integration with Google Cloud Platform,
  - Support for TPUs,

Source: https://deepmind.com/css/images/opengraph/alphago-logo.png

Use Cases

- Google DeepMind now using TensorFlow
- Already for *AlphaGo*
- According to a DeepMind SWE reasons are:
  - Python,
  - Integration with Google Cloud Platform,
  - Support for TPUs,
  - Ability to run on many GPUs.

Source: https://deepmind.com/css/images/opengraph/alphago-logo.png

Walkthrough

How do I continue?

Resources

Resources

- MOOCs

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity
- Websites

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity
- Websites
  - http://colah.github.io

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity
- Websites
  - http://colah.github.io
  - http://cs231n.github.io

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity
- Websites
  - http://colah.github.io
  - http://cs231n.github.io
  - http://karpathy.github.io

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity
- Websites
  - http://colah.github.io
  - http://cs231n.github.io
  - http://karpathy.github.io
  - http://www.deeplearningbook.org

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity
- Websites
  - http://colah.github.io
  - http://cs231n.github.io
  - http://karpathy.github.io
  - http://www.deeplearningbook.org
  - http://neuralnetworksanddeeplearning.com

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity
- Websites
  - http://colah.github.io
  - http://cs231n.github.io
  - http://karpathy.github.io
  - http://www.deeplearningbook.org
  - http://neuralnetworksanddeeplearning.com
  - https://www.kaggle.com

Resources

- MOOCs
  - Machine Learning by Andrew Ng @ Coursera
  - Deep Learning by Google @ Udacity
  - Machine Learning Nanodegree @ Udacity
- Websites
  - http://colah.github.io
  - http://cs231n.github.io
  - http://karpathy.github.io
  - http://www.deeplearningbook.org
  - http://neuralnetworksanddeeplearning.com
  - https://www.kaggle.com
  - https://www.tensorflow.org

Deep Learning in Action #4

ACM Munich Student Chapter

Meetups

Deep Learning in Action #4

ACM Munich Student Chapter

- Monday, August 1, 2016

Deep Learning in Action #4

ACM Munich Student Chapter

- Monday, August 1, 2016
- Geometric Deep Learning by Prof. Dr. Michael M. Bronstein

Deep Learning in Action #4

ACM Munich Student Chapter

- Monday, August 1, 2016
- Geometric Deep Learning by Prof. Dr. Michael M. Bronstein
- Introduction to TensorFlow

Meetups

Deep Learning in Action #4

ACM Munich Student Chapter

- Monday, August 1, 2016
- Geometric Deep Learning by Prof. Dr. Michael M. Bronstein
- Introduction to TensorFlow
- More to come

Stay in Touch

Stay in Touch

- peter@goldsborough.me

Stay in Touch

- peter@goldsborough.me
- linkedin.com/in/petergoldsborough

Stay in Touch

- peter@goldsborough.me
- linkedin.com/in/petergoldsborough
- github.com/goldsborough

Stay in Touch

- peter@goldsborough.me
- linkedin.com/in/petergoldsborough
- github.com/goldsborough
- @peterawks

Q & A

References


Pedro Domingos, *A few useful things to know about machine learning*, Commun. ACM **55** (2012), no. 10, 78–87.


Andrej Karpathy, *The unreasonable effectiveness of recurrent neural networks*, May 21 2015 (accessed Jul 10, 2016), http://karpathy.github.io/2015/05/21/rnn-effectiveness/.


_____, *Neural networks part 1: Setting up the architecture*, CS231n: Convolutional Neural Networks for Visual Recognition, 2016 (accessed July 9, 2016), http://cs231n.github.io/neural-networks-1/.


Thomas M. Mitchell, *Machine learning*, 1 ed., McGraw-Hill, Inc., New York, NY, USA, 1997.


New navy device learns by doing, July 08 1958 (accessed Jul 9, 2016), http://query.nytimes.com/gst/abstract.html?res=9D01E4D8173DE53BBC4