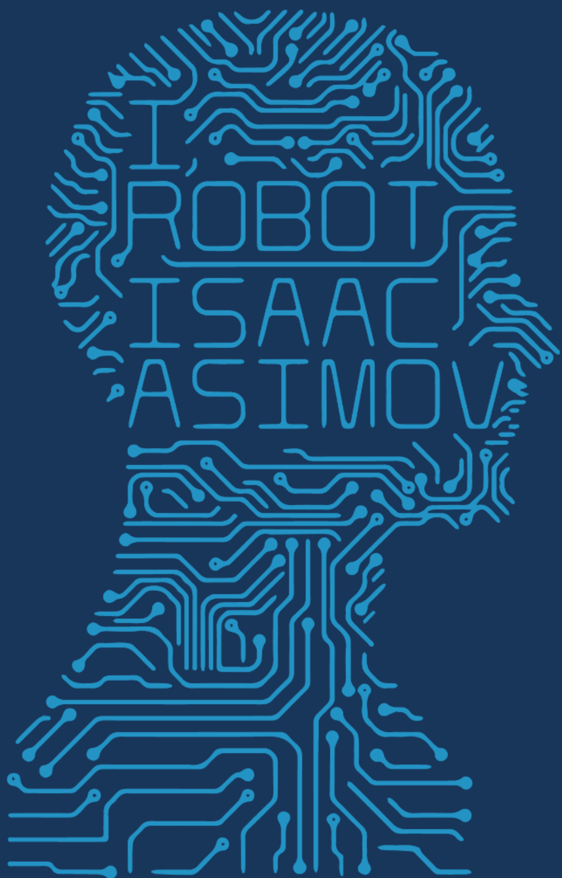
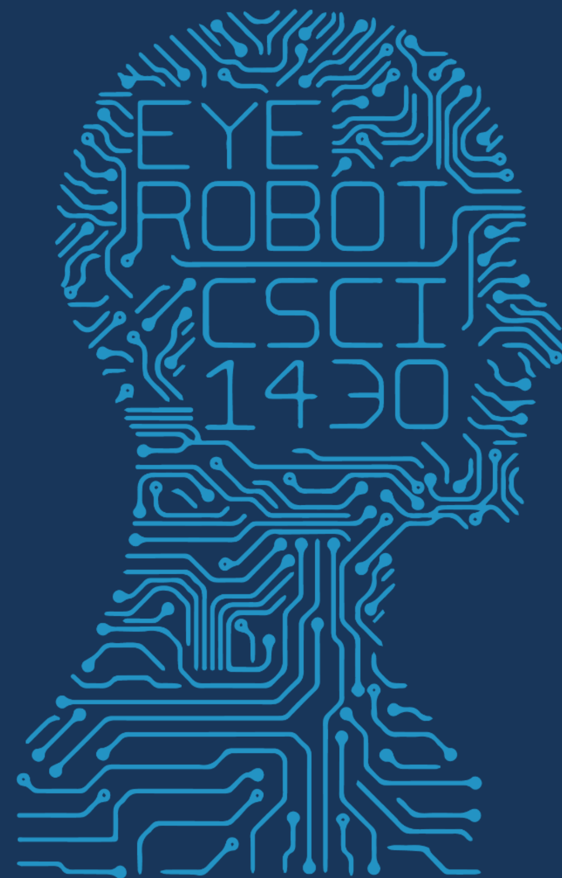


1950

Future Vision



2017 MWF 1pm 368

Computer Vision

Warning – new jargon

- Learning jargon is always painful…

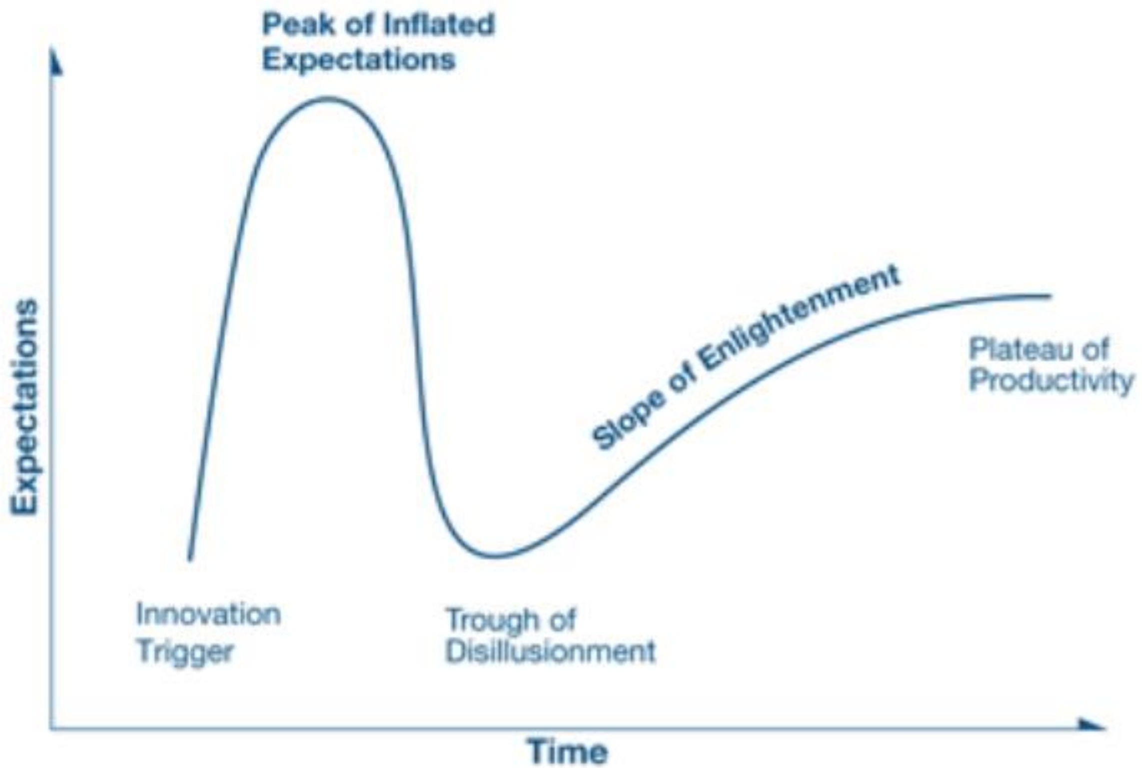
…even if the concepts behind the jargon are not hard.

So, let’s get used to it.

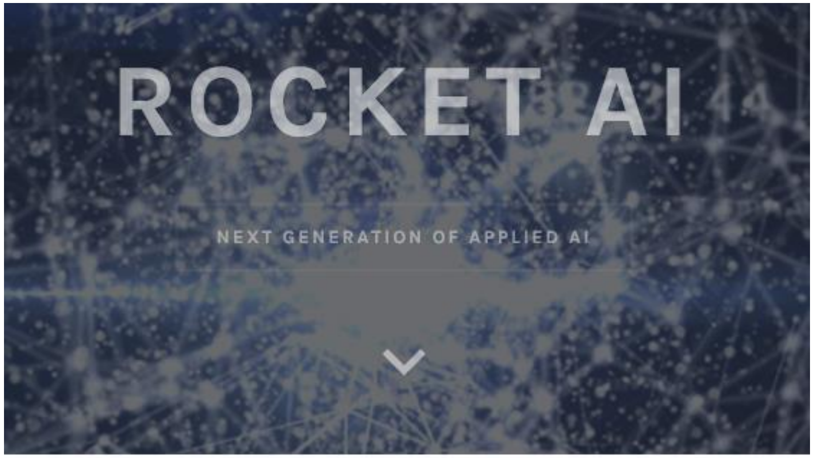
“In mathematics you don't understand things.
You just get used to them.”

von Neumann (a joke)

Gartner Hype Cycle



Rocket AI



Launching in 2017, Rocket AI will be the global leader in neurologically-inspired applied machine learning.

We build our systems around our patent-pending technology

Temporally Recurrent Optimal Learning™

We Are Hiring

launch@rocketai.org

Rocket AI

- Launch party @ NIPS 2016
- Neural Information Processing Systems
- Academic conference

Markus Wulfmeier

December 8 at 5:15pm · Barcelona, Spain ·

#rocketai s launch party at #nips2016 clearly the best. Including the police involvement.

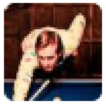
Rocket AI

Pôle #AI au #Québec and 22 others liked

Andrej Karpathy @karpathy · Dec 9

Best party of #nips2016 award goes to **#rocketai** (rocketai.org). Definitely a company to watch closely.

5 44 130



Karl Moritz Hermann @karlmoritz · Dec 9

One day we will look back and realise that the **#rocketai** launch was the day when things in our field changed forever.

2 3 20

Ian Goodfellow @goodfellow_ian · Dec 11

#rocketai definitely has the most popular Jacobian-Optimized Kernel Expansion of NIPS 2016

6 42 214

Rocket AI

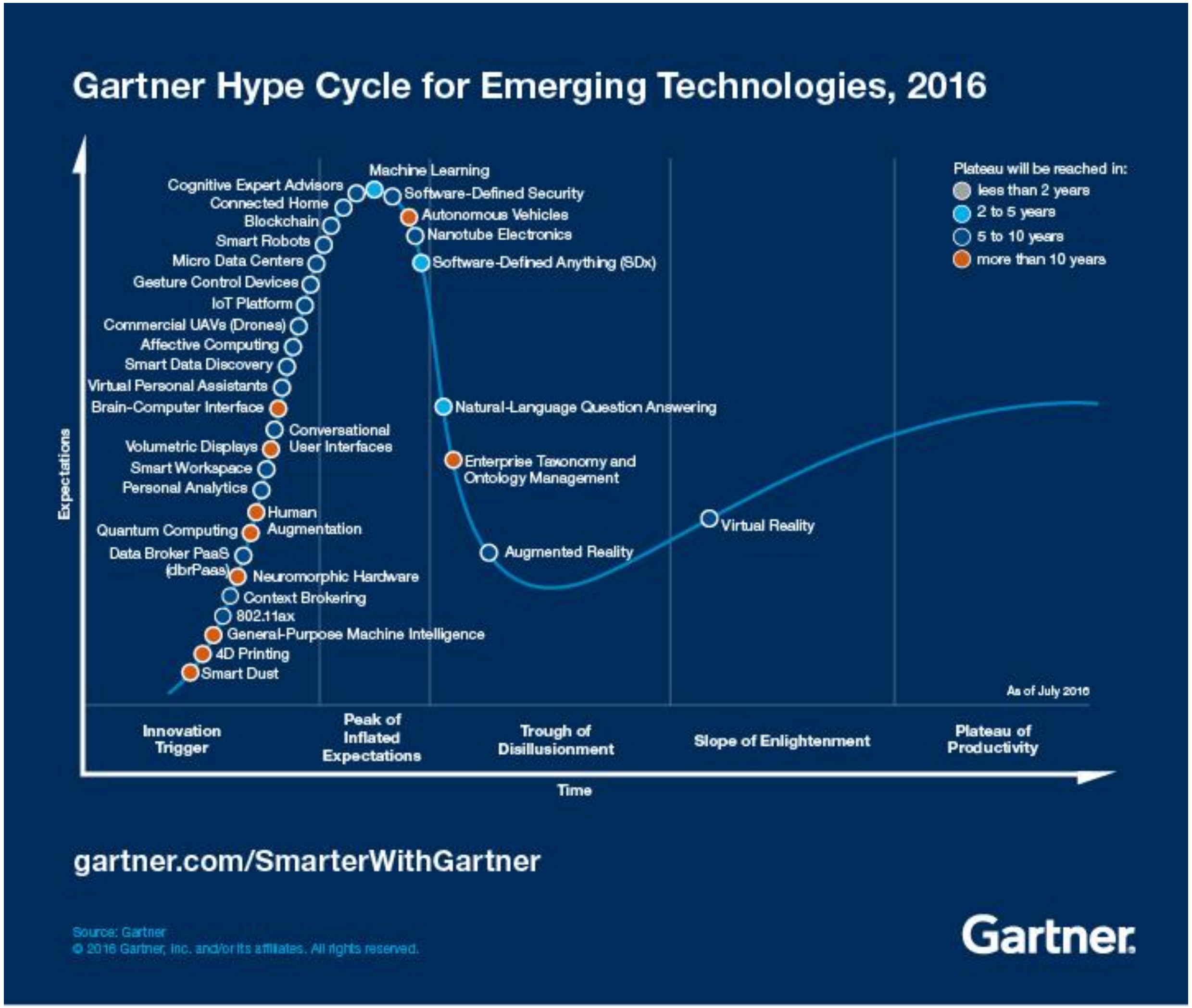
Metrics for the Rocket AI launch party

Email RSVPs to party: 316
People who emailed in their resume: 46
Large name brand funds who contacted us about investing: 5
Media: Twitter, Facebook, HackerNews, Reddit, Quora, Medium etc
Time Planning: < 8 hours
Money Spent: $79 on the domain, $417 on alcohol and snacks + (police fine)
For reference, NIPS sponsorship starts at $10k.

Estimated value of Rocket AI: *in the tens of millions.*

Article by Riva-Melissa Tez

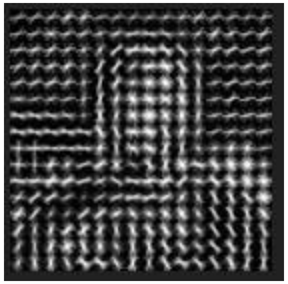
Gartner Hype Cycle



So far…

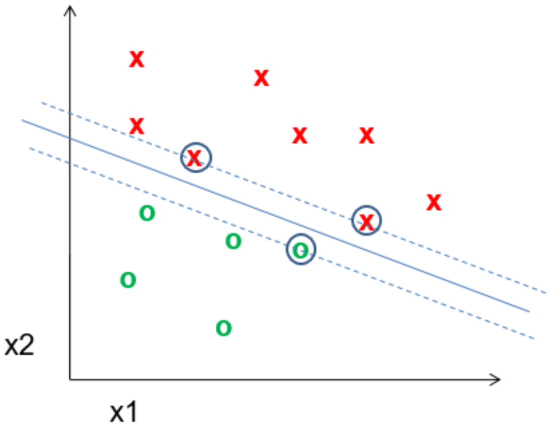
Best performing visions systems have commonality:

- Hand designed features
  - Gradients + non-linear operations (exponentiation, clamping, binning)
  - Features in combination (parts-based models)
  - Multi-scale representations



- Machine learning from databases

- Linear classifiers (SVM)



But it’s still not that good…

- PASCAL VOC = ~75%
- ImageNet = ~75%; human performance = ~95%

Previous claim:

It is more important to have
more or better labeled data than to use
a different supervised learning technique.

“The Unreasonable Effectiveness of Data” - Norvig

No free lunch theorem

Hume (c.1739):

“‘Even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience.”

-> Learning beyond our experience is impossible.

No free lunch theorem

Wolpert (1996):

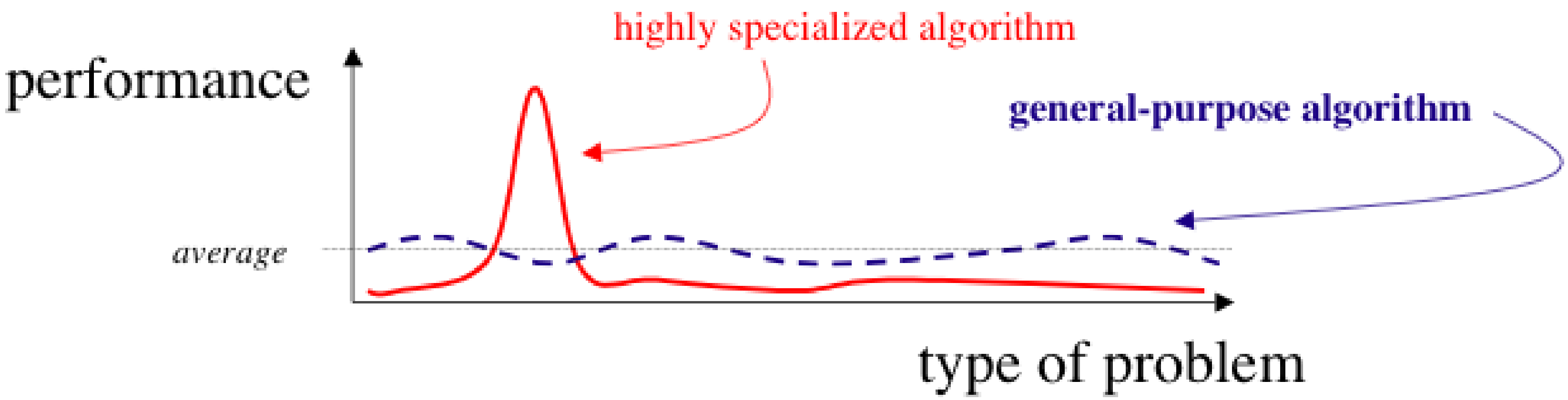
‘No free lunch’ for supervised learning:

“In a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no *a priori* distinctions between learning algorithms.”

-> Averaged over all possible datasets, no learning algorithm is better than any other.

OK, well, let's give up. Class over.

No, no, no!



We can build a classifier which better matches the characteristics of the problem!

But…didn’t we just do that?

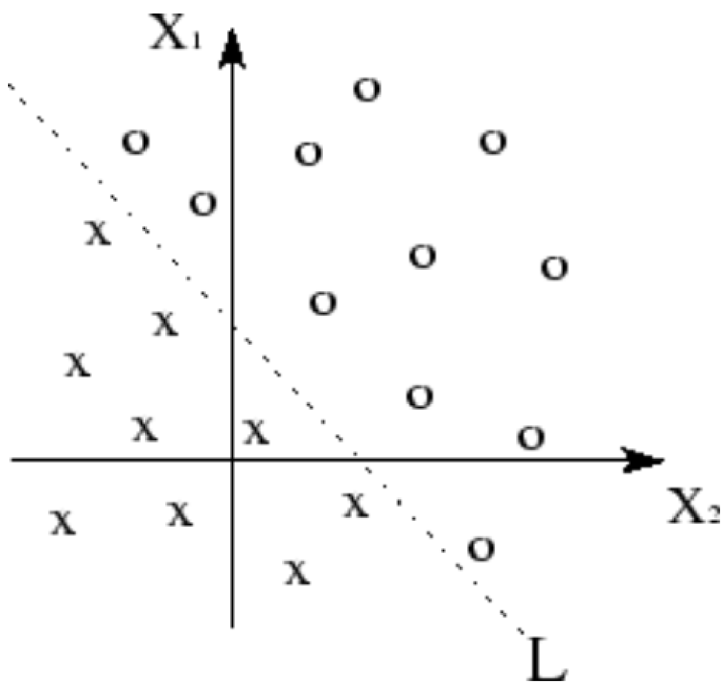
- PASCAL VOC = ~75%
- ImageNet = ~75%; human performance = ~95%

We used intuition and understanding of how we think vision works, but it still has limitations.

Why?

Linear spaces - separability

- + kernel trick to transform space.



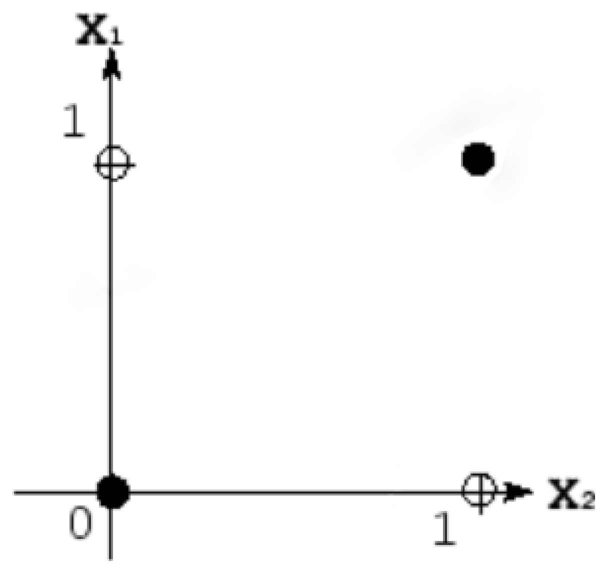
Linearly separable data
+ linear classifer = good.

Non-linear spaces - separability

- Take XOR – exclusive OR
- E.G., human face has two eyes XOR sunglasses

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$Y = X_1 \oplus X_2$$

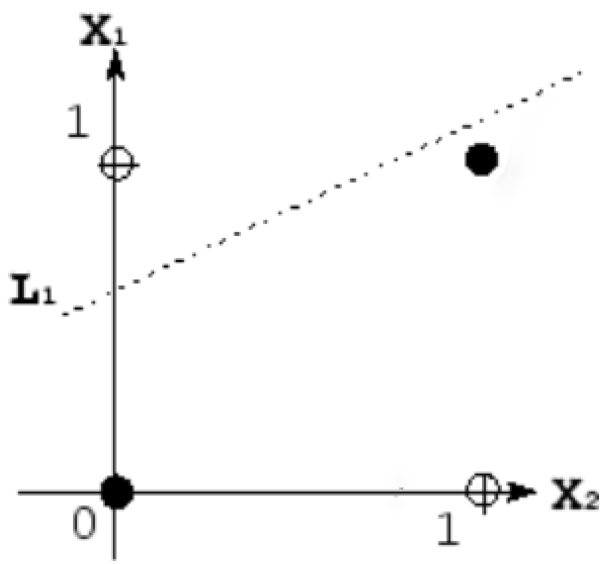


Non-linear spaces - separability

- Linear functions are insufficient on their own.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$Y = X_1 \oplus X_2$$

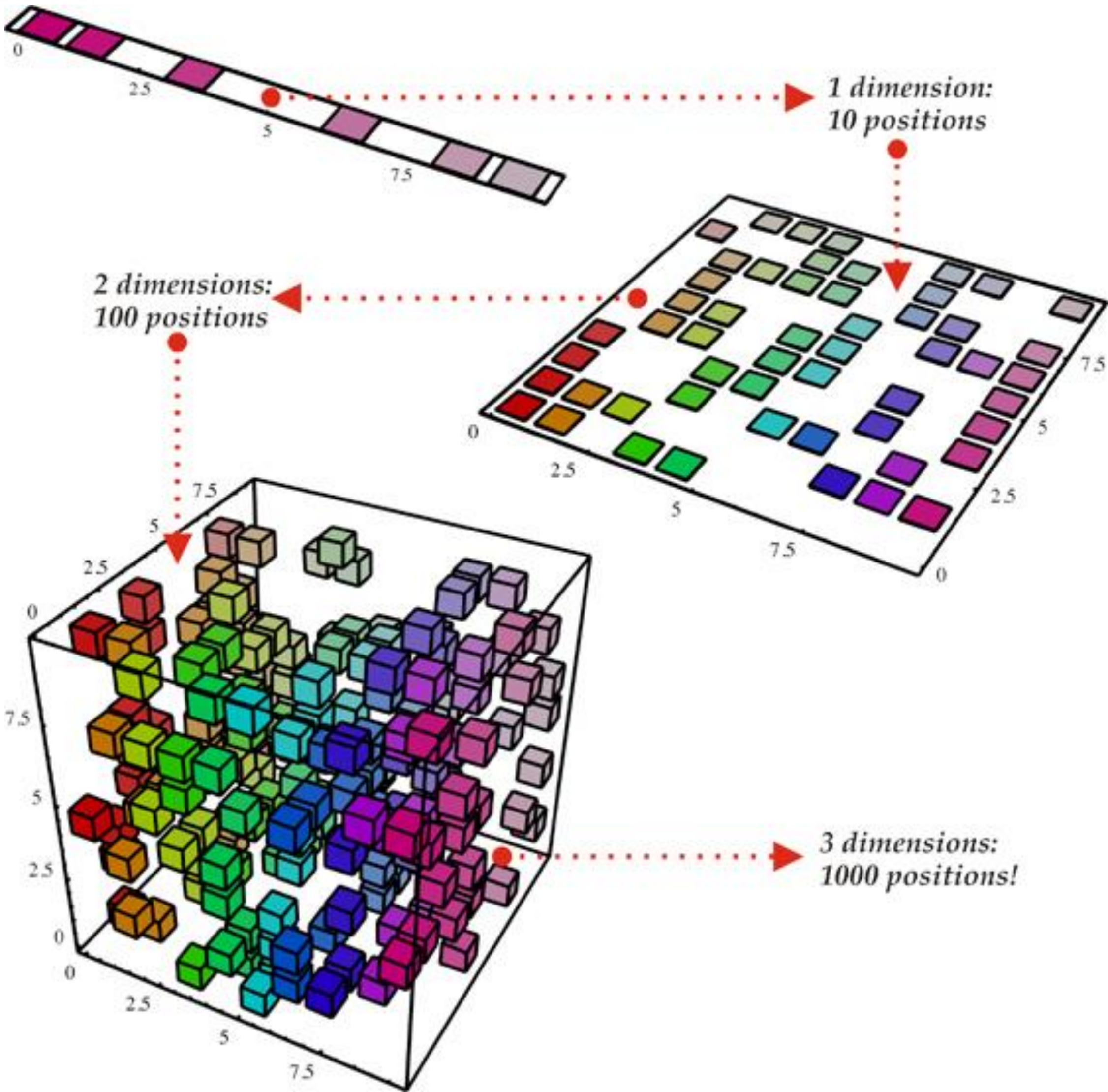


Kawaguchi

Curse of Dimensionality

Every feature that we add requires us to learn the useful regions in a much larger volume.

d binary variables = $O(2^d)$ combinations

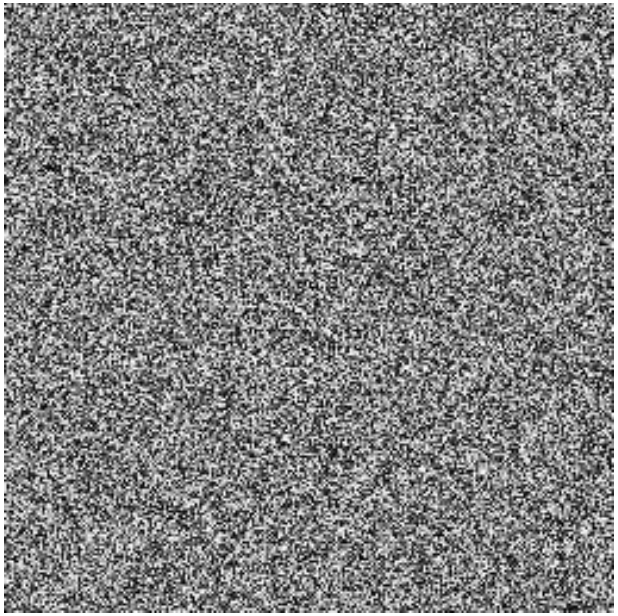


Curse of Dimensionality

- Not all regions of this high-dimensional space are meaningful.

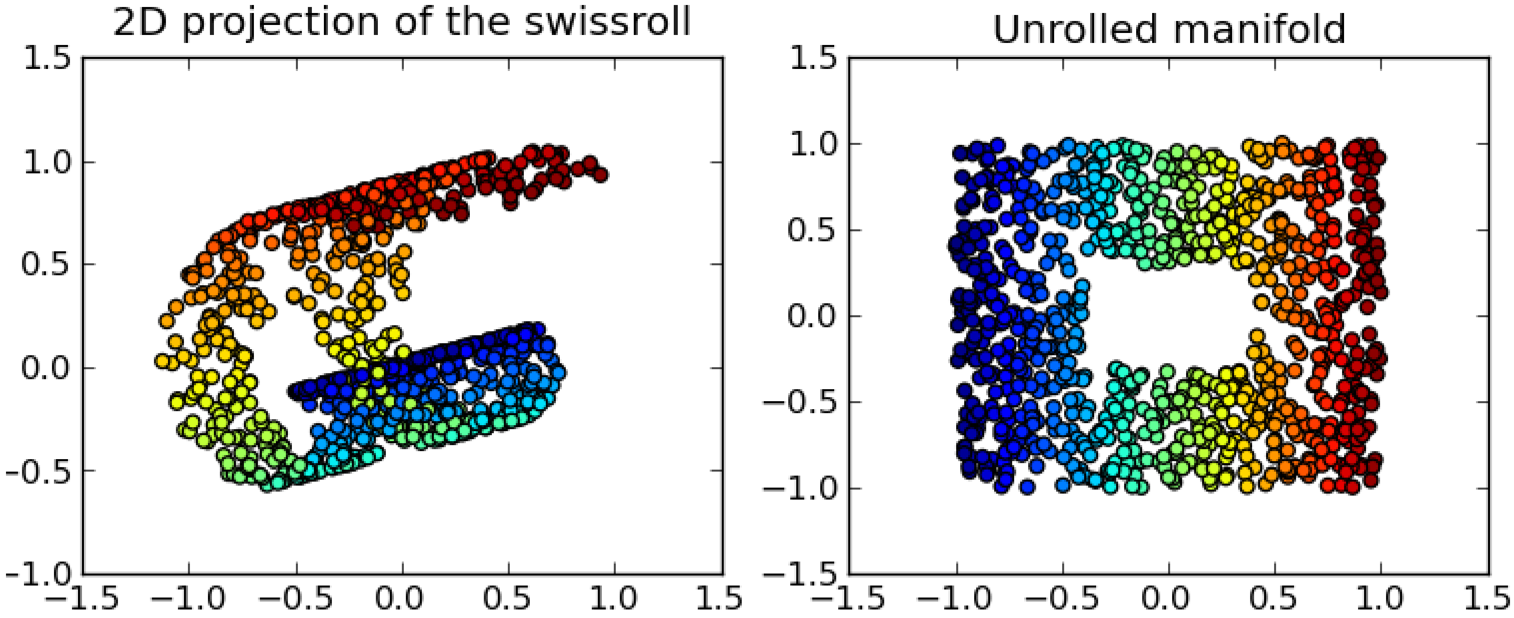
```
>> I = rand(256,256);
>> imshow(I);
```

@ 8bit = 256 values ^ 65,536



Related: Manifold Learning

Learning locally low-dimensional Euclidean spaces connected and embedded within a high-dim. space.



Wikipedia

Local constancy / smoothness of feature space

- All existing learning algorithms we have seen assume **smoothness** or **local constancy.**
  - -> New example will be near existing examples
  - -> Each region in feature space requires an example

Smoothness is ‘averaging’ or ‘interpolating’.

Local constancy / smoothness of feature space

- At the extreme: Take k-NN classifier.
- The number of regions cannot be more than the number of examples.

-> No way to generalize beyond examples

How to represent a complex function with *more factors* than regions?

(Deep) Neural Networks

Goals

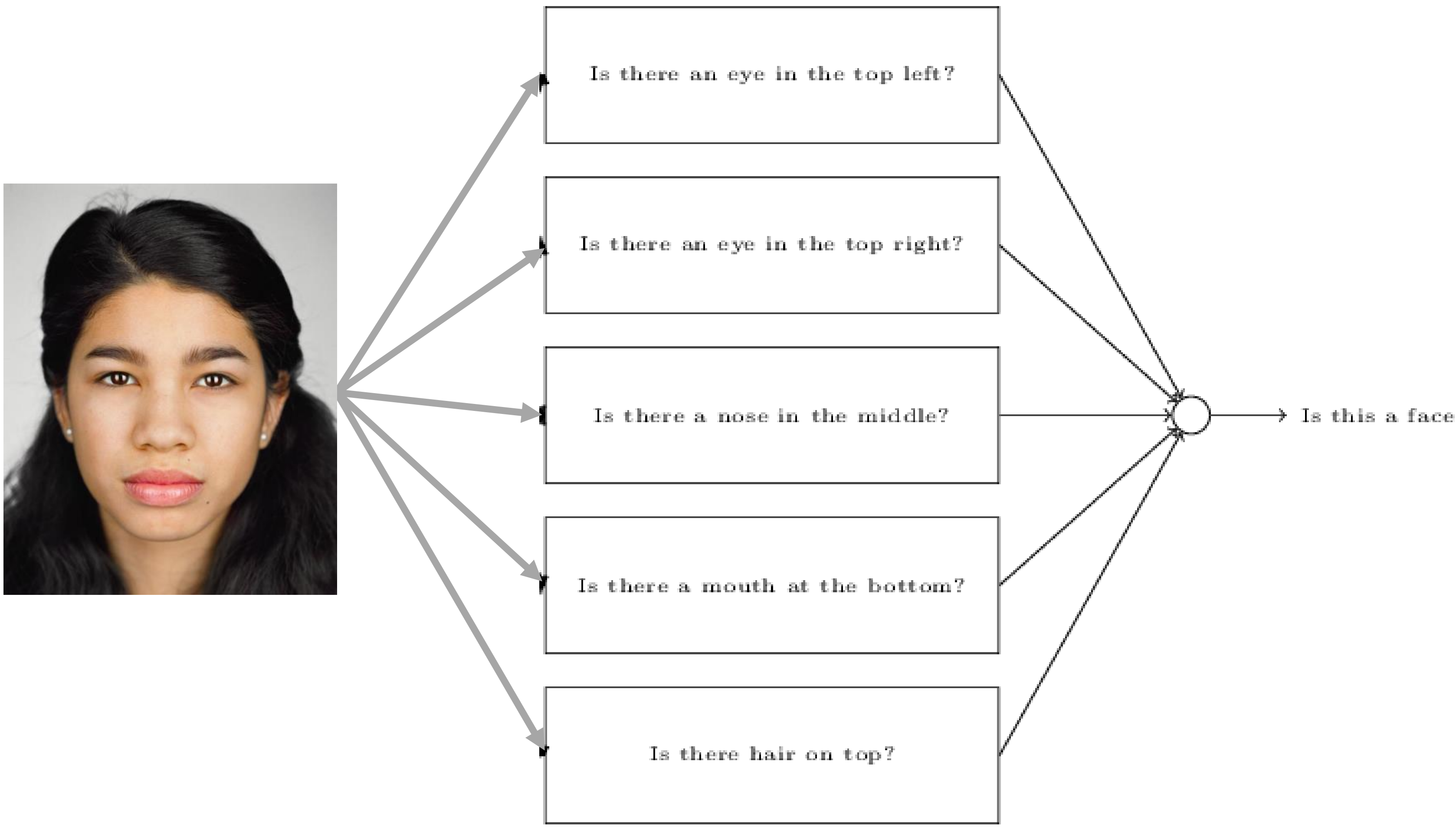
- Build a classifier which is more powerful at representing complex functions *and* more suited to the learning problem.

What does this mean?

1. Assume that the *underlying data generating function* relies on a composition of factors in a hierarchy.

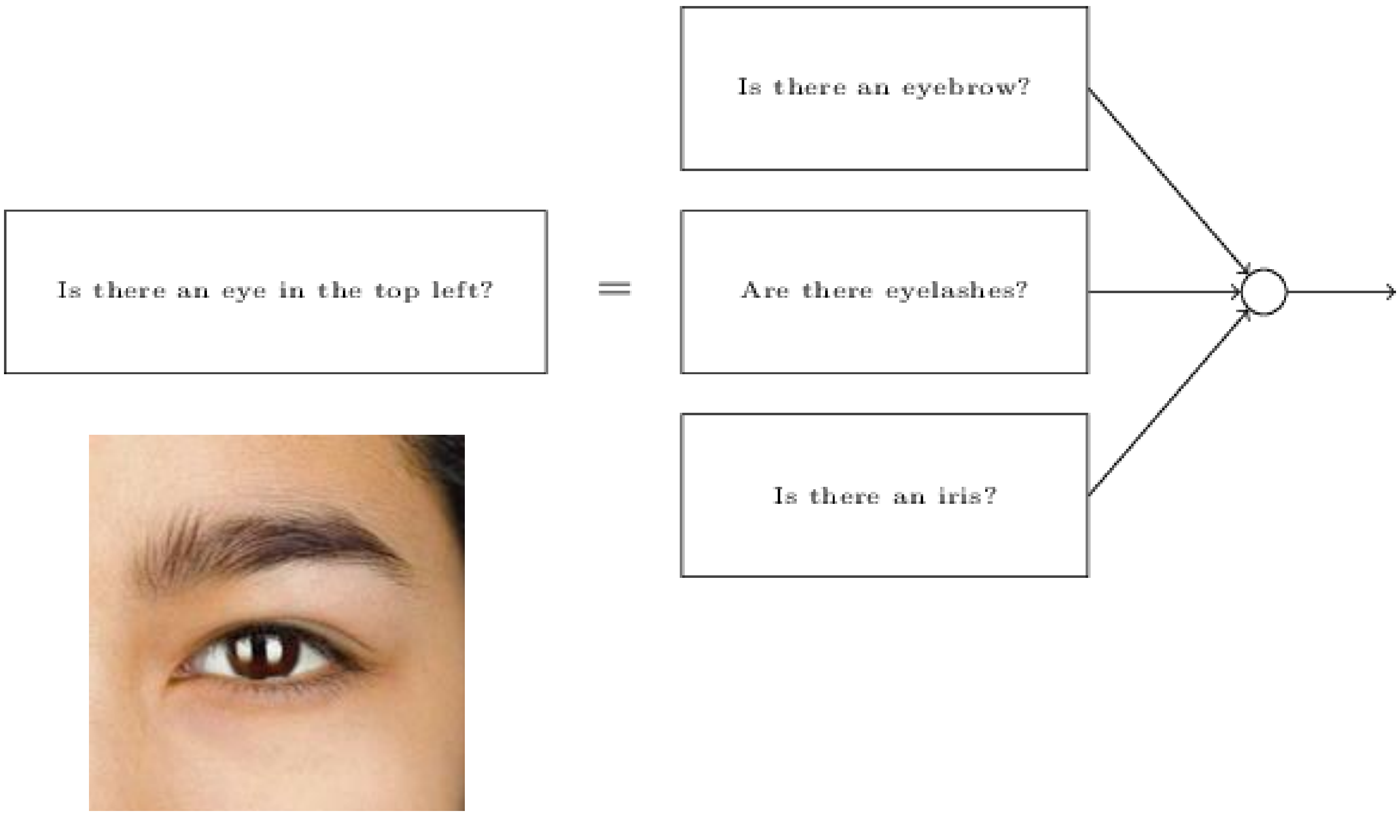
Factor composition = dependencies between regions in feature space.

Example



Nielsen, National Geographic

Example

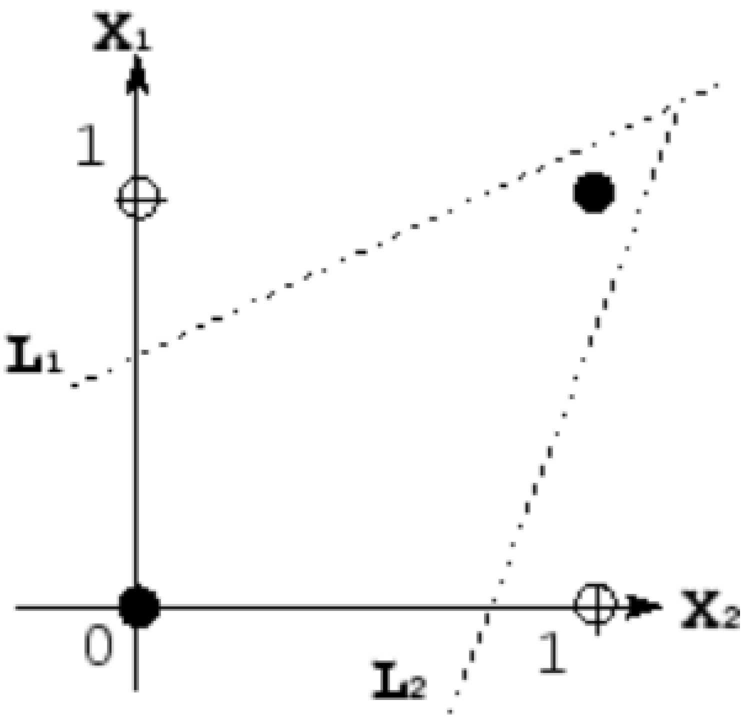

Is there an eye in the top left?
=
Is there an eyebrow?
Are there eyelashes?
Is there an iris?

Non-linear spaces - separability

- *Composition* of linear functions can represent more complex functions.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$Y = X_1 \oplus X_2$$



Kawaguchi

Goals

- Build a classifier which is more powerful at representing complex functions *and* more suited to the learning problem.

What does this mean?

2. Learn a feature representation that is specific to the dataset.

10k/100k+ data points + factor composition = sophisticated representation.

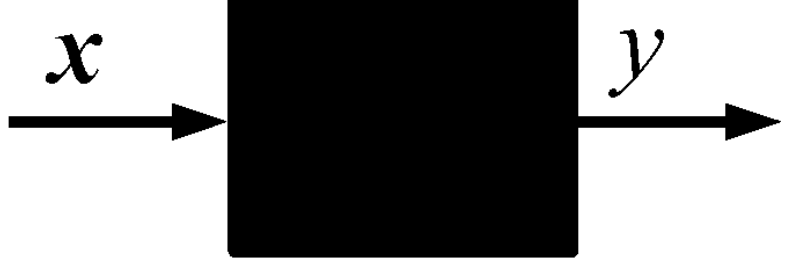
Supervised Learning

$\left\{ (\boldsymbol{x}^i, y^i), i=1 \ldots P \right\}$ training dataset

$\boldsymbol{x}^i$ i-th input training example

y^i i-th target label

P number of training examples



Goal: predict the target label of unseen inputs.

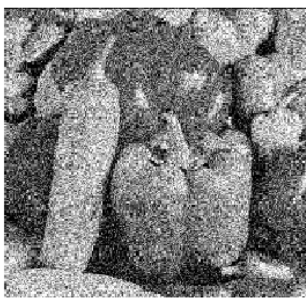
Supervised Learning: Examples

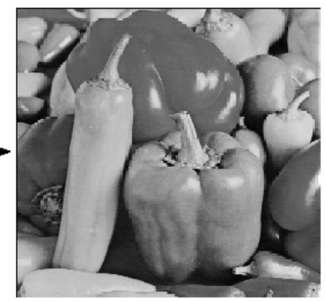
Classification

“dog”

classification

Denoising





regression

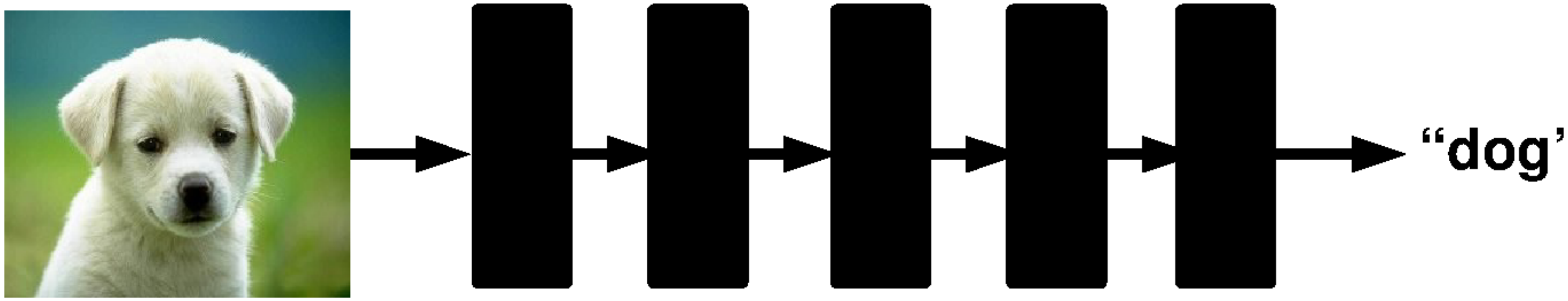
OCR

“2 3 4 5”

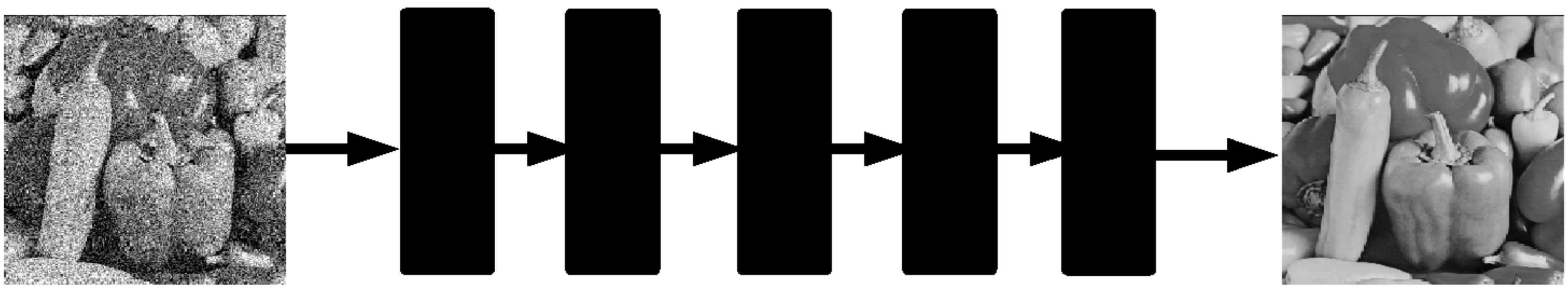
structured prediction

Supervised Deep Learning

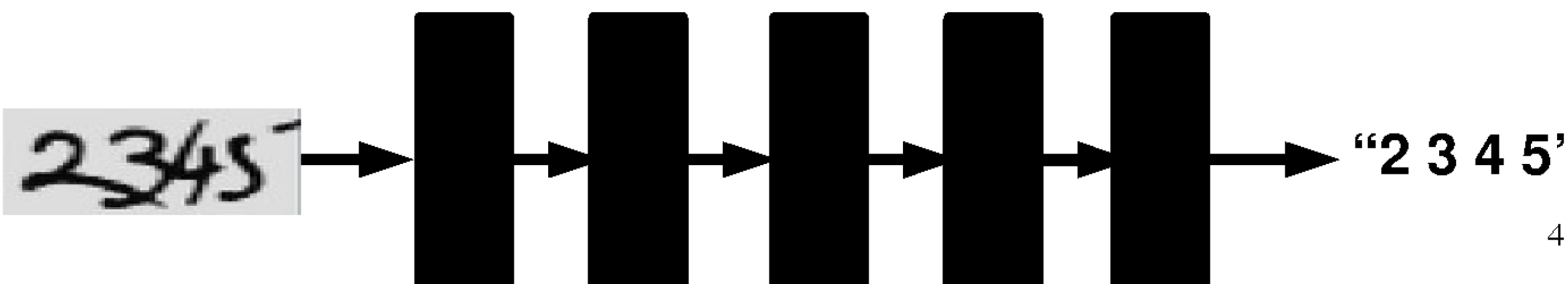
Classification



Denoising

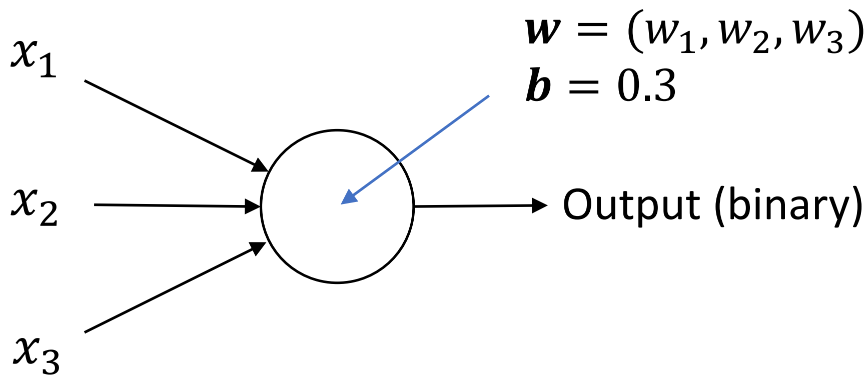


OCR



Neural Networks

- Basic building block for composition is a *perceptron* (Rosenblatt c.1960)
- Linear classifier – vector of weights *w* and a 'bias' *b*



$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \qquad w \cdot x \equiv \sum_j w_j x_j$$

Binary classifying an image

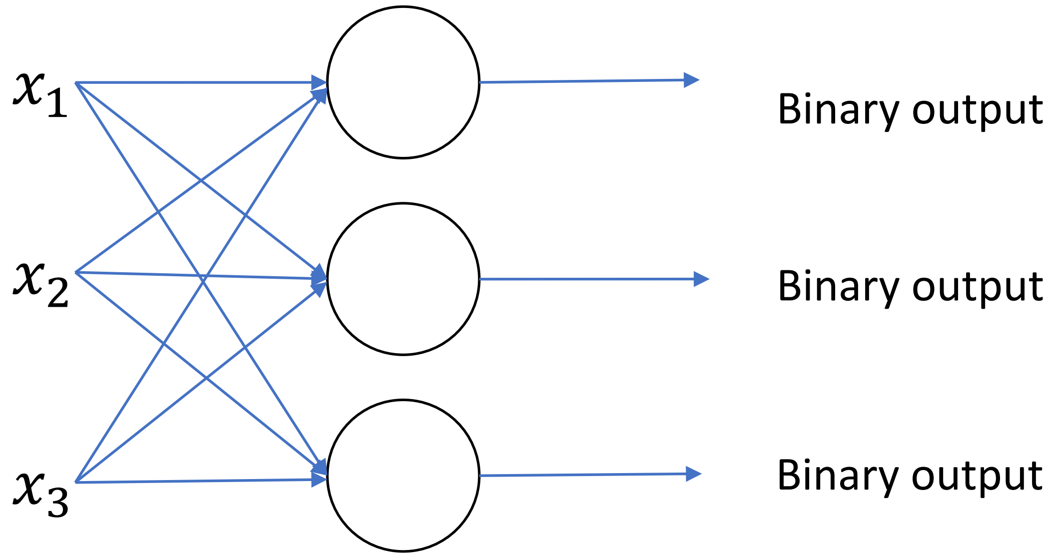
- Each pixel of the image would be an input.
- So, for a 28 x 28 image, we vectorize.
- **x** = 1 x 784

- **w** is a vector of weights for each pixel, 784 x 1
- b is a scalar bias per perceptron

- result = **xw** + b   -> (1x784) x (784x1) + b = (1x1)+b

Neural Networks - multiclass

- Add more perceptrons



Multi-class classifying an image

- Each pixel of the image would be an input.
- So, for a 28 x 28 image, we vectorize.
- **x** = 1 x 784

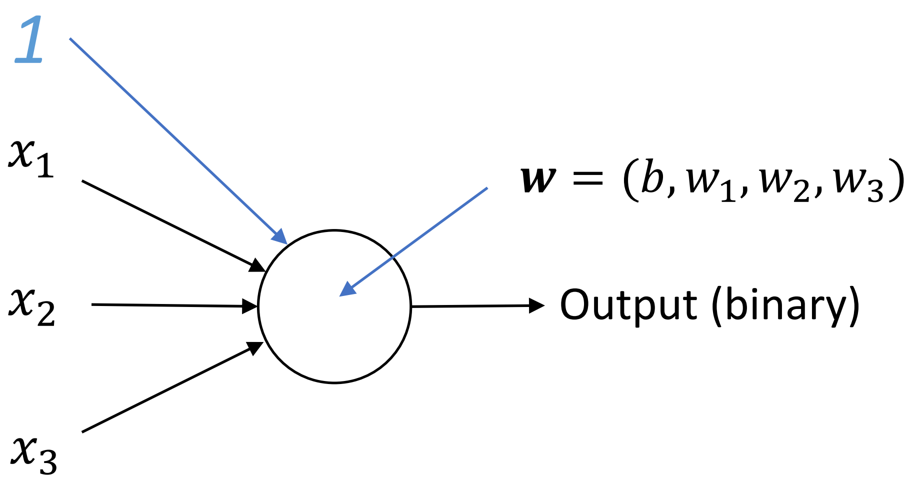
- **W** is a matrix of weights for each pixel/each perceptron
  - **W** = 10 x 784 (10-class classification)
- **b** is a bias *per perceptron* (vector of biases); (1 x 10)

- result = **xW** + **b**   -> (1x784) x (784 x 10) + **b**

  -> (1 x 10) + (1 x 10) = output vector

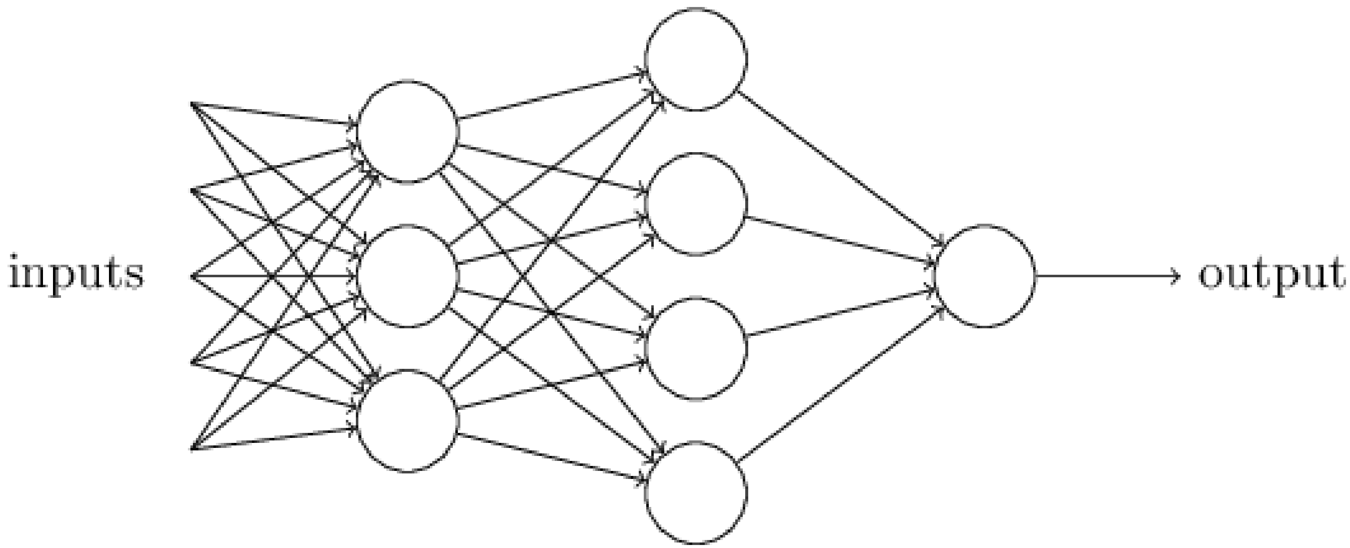
Bias convenience

- To turn this classification operation into a multiplication only:
  - Create a ‘fake’ feature with value 1 to represent the bias
  - Add an extra weight that can vary



$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x \quad \leq 0 \\ 1 & \text{if } w \cdot x \quad > 0 \end{cases} \qquad w \cdot x \equiv \sum_j w_j x_j,$$

Composition

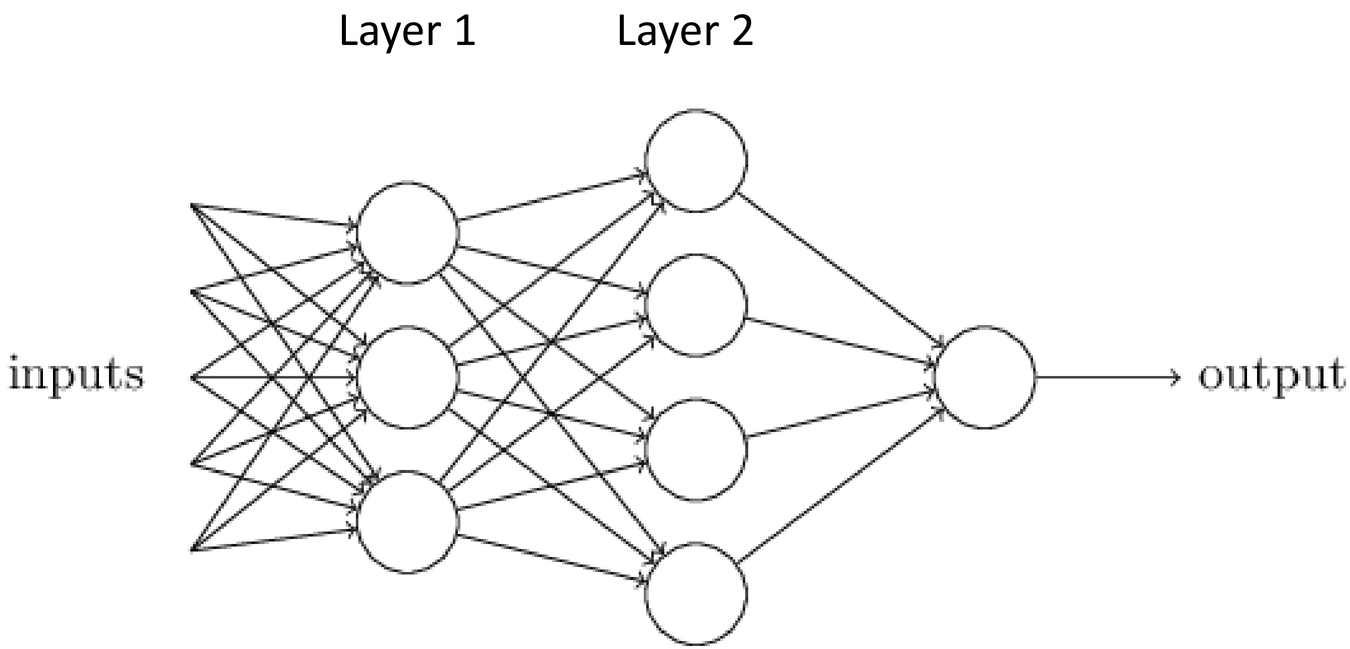


Attempt to represent complex functions as compositions of smaller functions.

Outputs from one perception are fed into inputs of another perceptron.

Nielsen

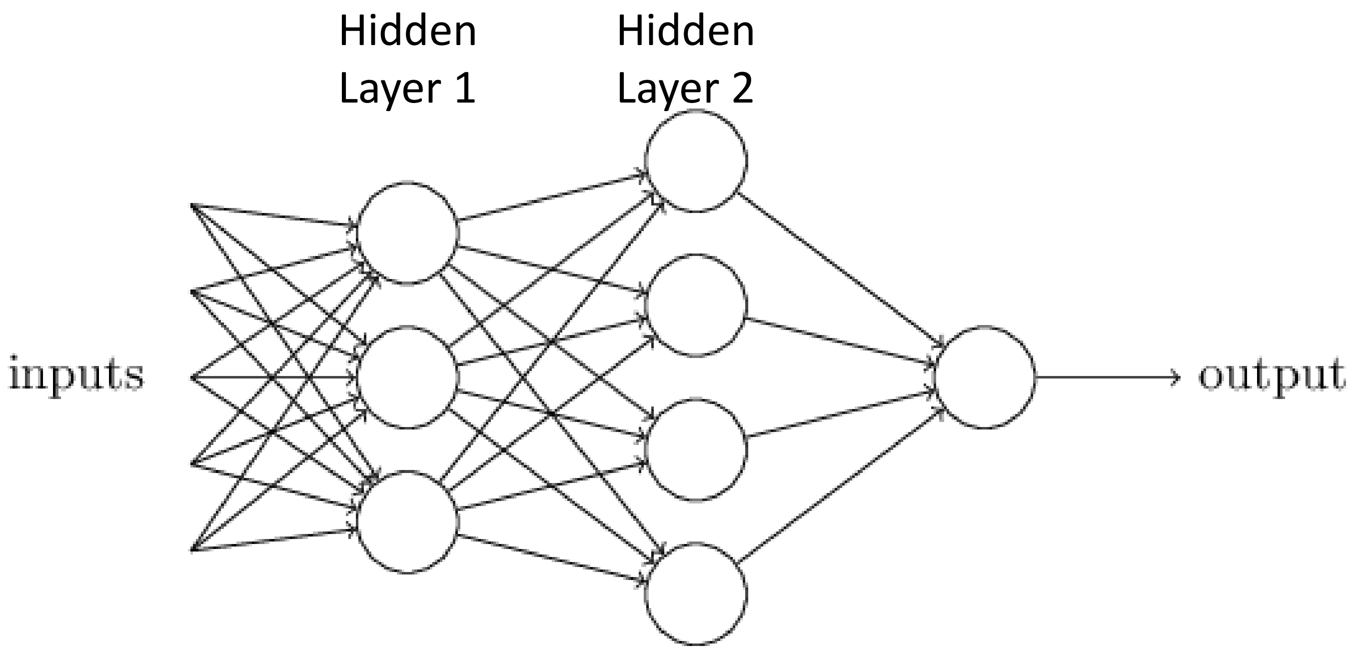
Composition



Sets of layers and the connections (weights) between them define the *network architecture.*

Nielsen

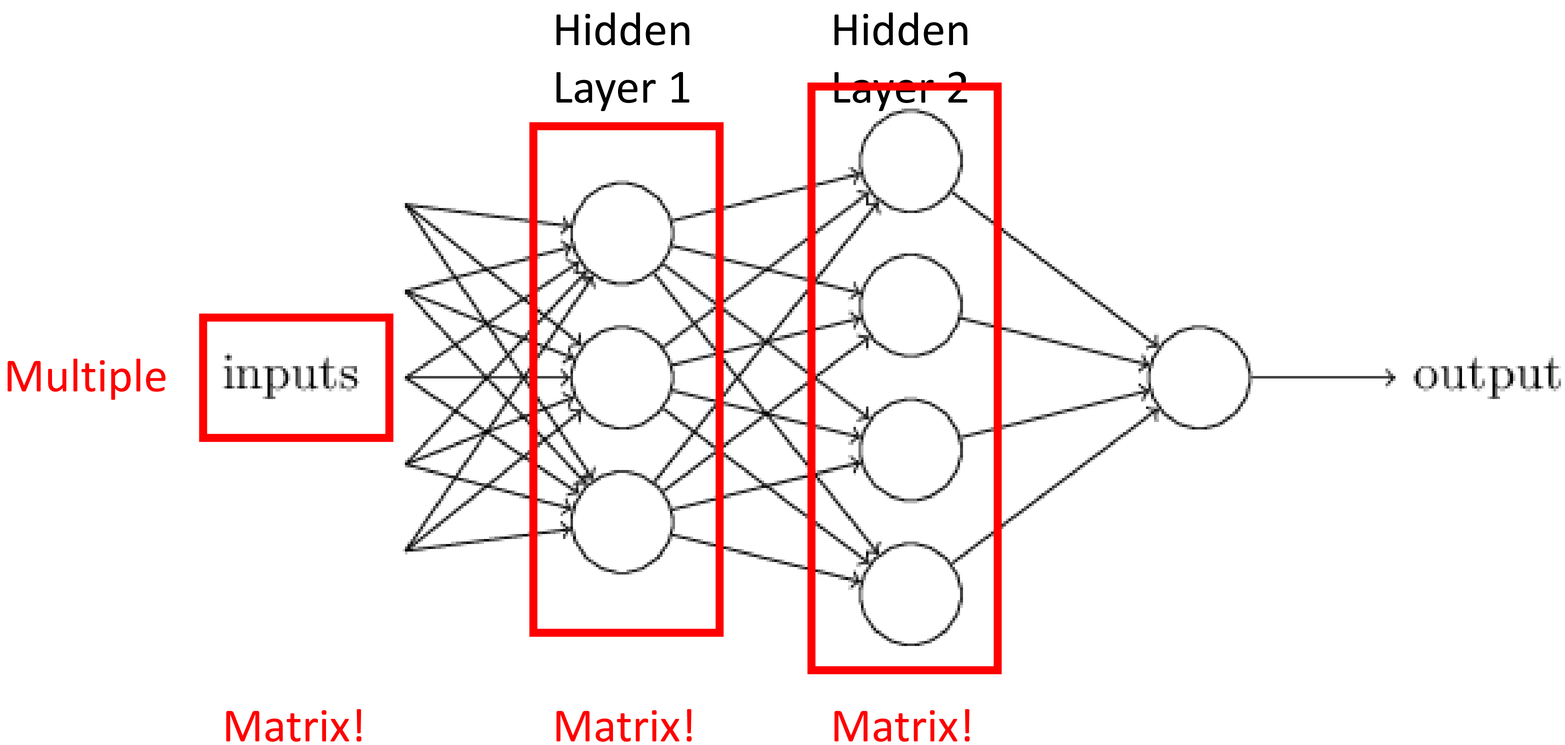
Composition



Layers that are in between the input and the output are called *hidden layers*, because we are going to *learn* their weights via an optimization process.

Nielsen

Composition



It's all just matrix multiplication!

GPUs -> special hardware for fast/large matrix multiplication.

Nielsen

Problem 1 with all linear functions

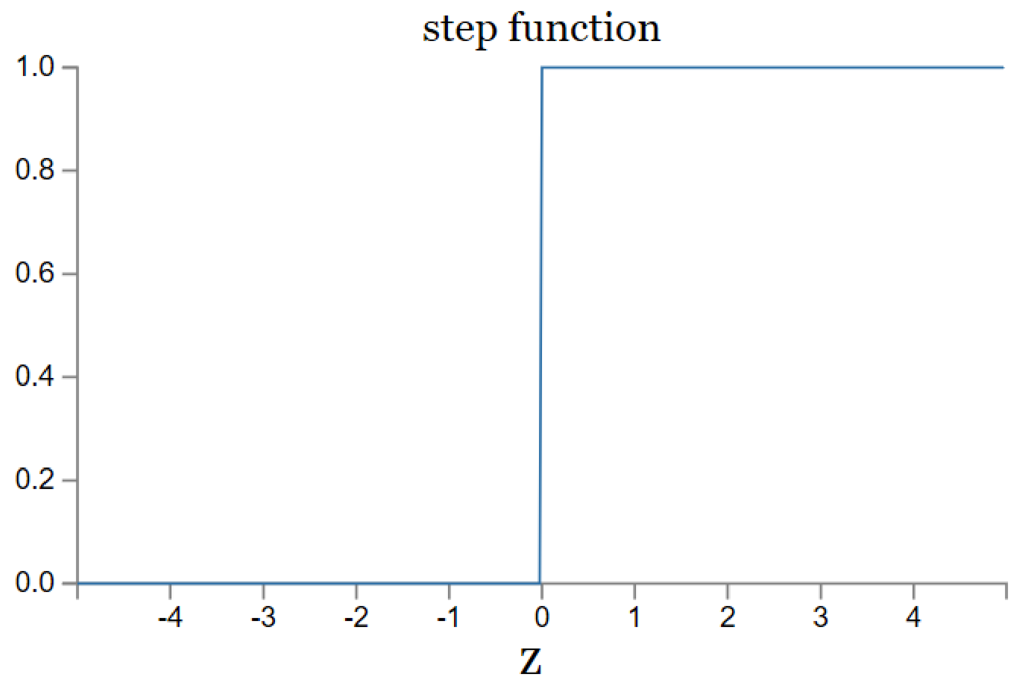
- We have formed chains of linear functions.
- We know that linear functions can be reduced
  - g = f(h(x))

- Our composition of functions is really just a single function : (

Problem 2 with all linear functions

- Linear classifiers: small change in input can cause large change in binary output.

Activation function

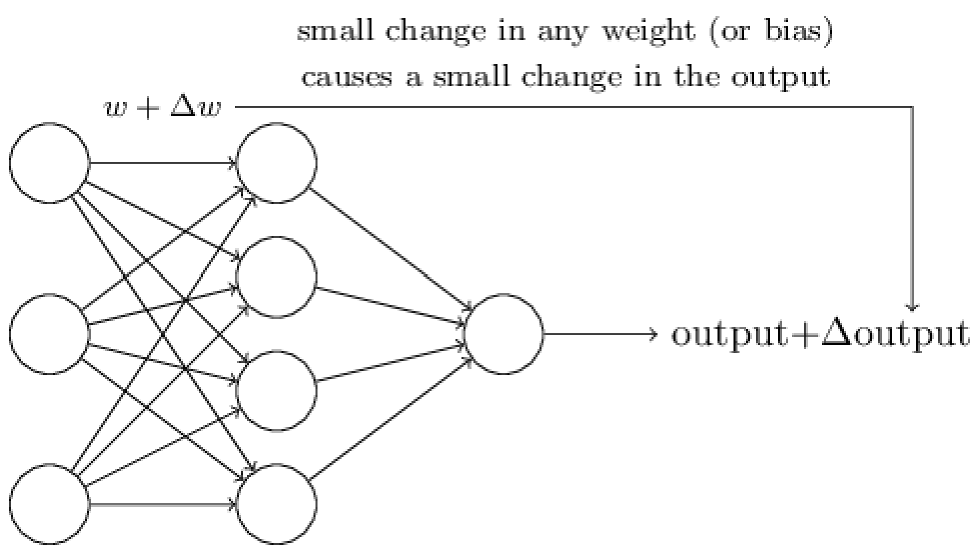


Nielsen

Problem 2 with all linear functions

- Linear classifiers: small change in input can cause large change in binary output.

- We want:



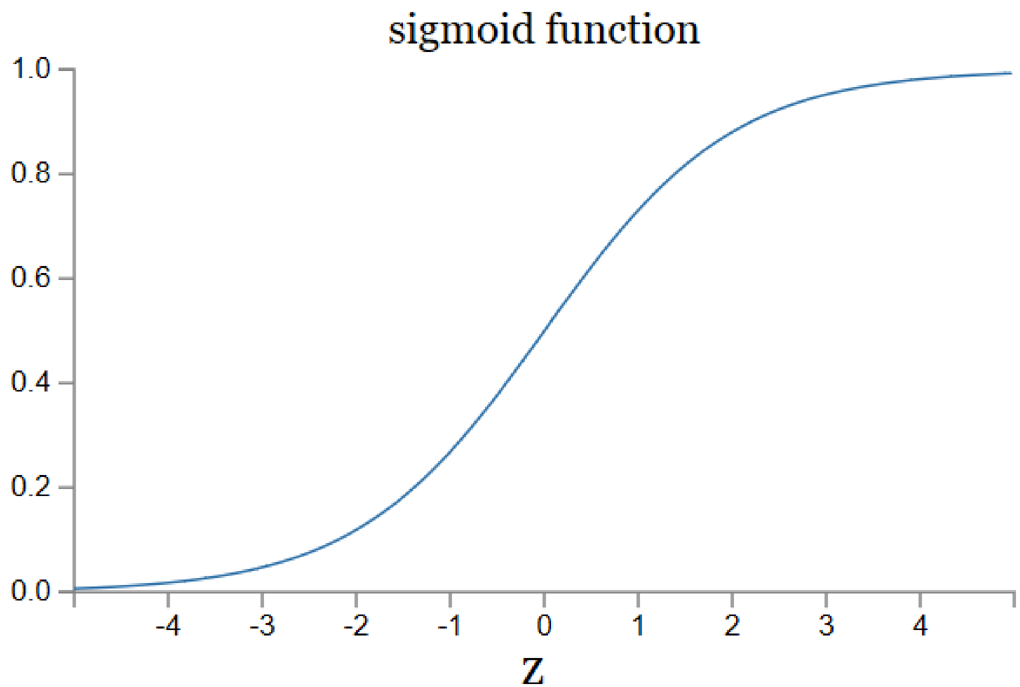
Nielsen

Let's introduce non-linearities

- We're going to introduce non-linear functions to transform the features.

$$\sigma(w \cdot x + b)$$

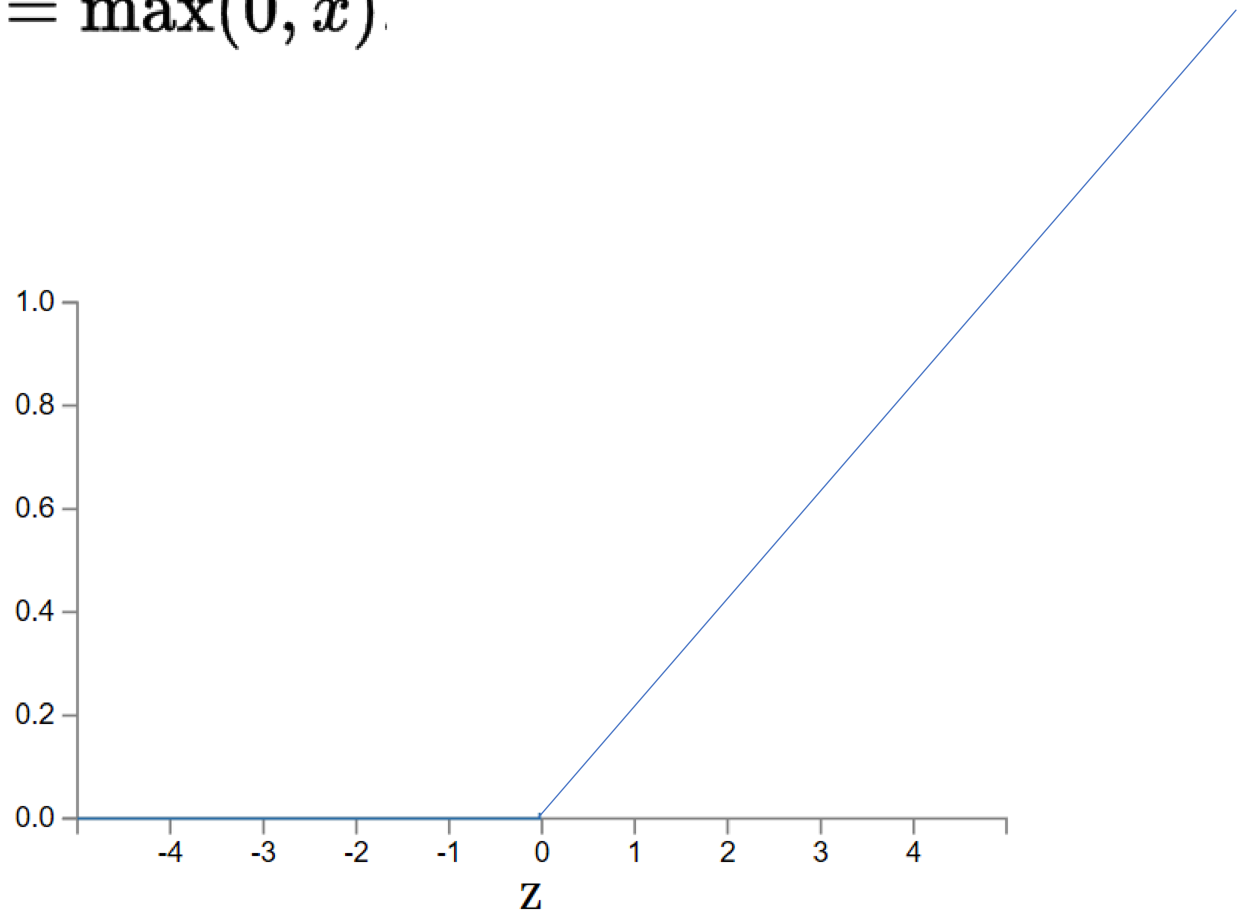
$$\sigma(z) \equiv \frac{1}{1+e^{-z}}.$$

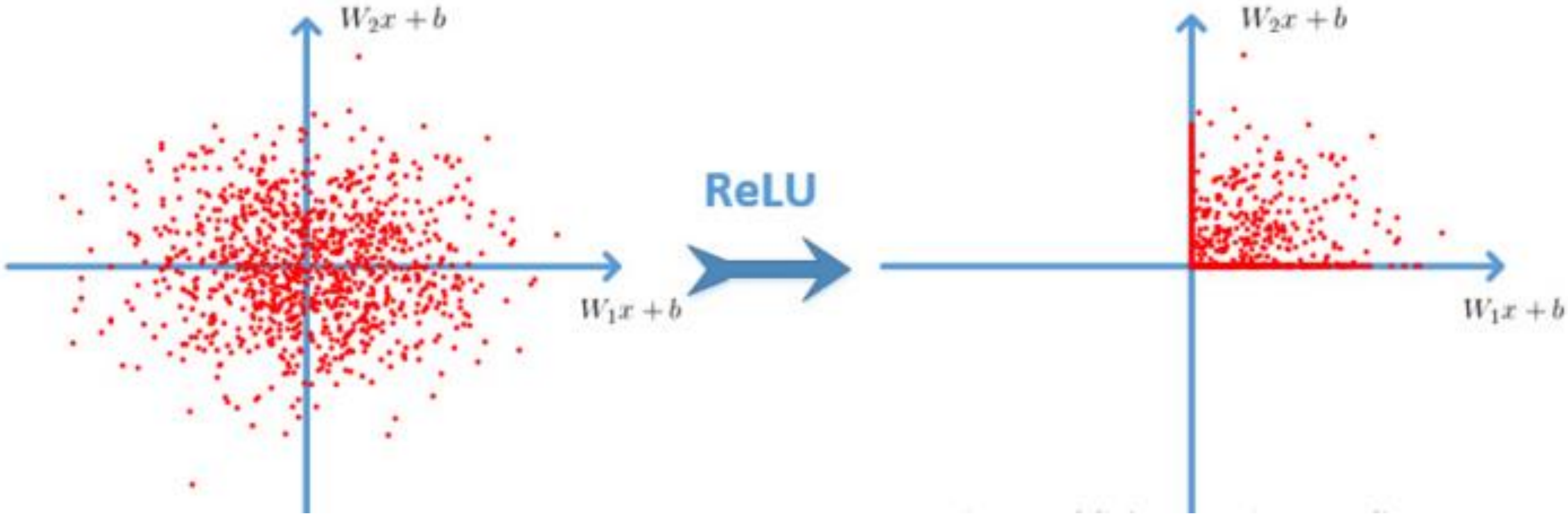


Nielsen

Rectified Linear Unit

- ReLU $f(x) = \max(0, x)$

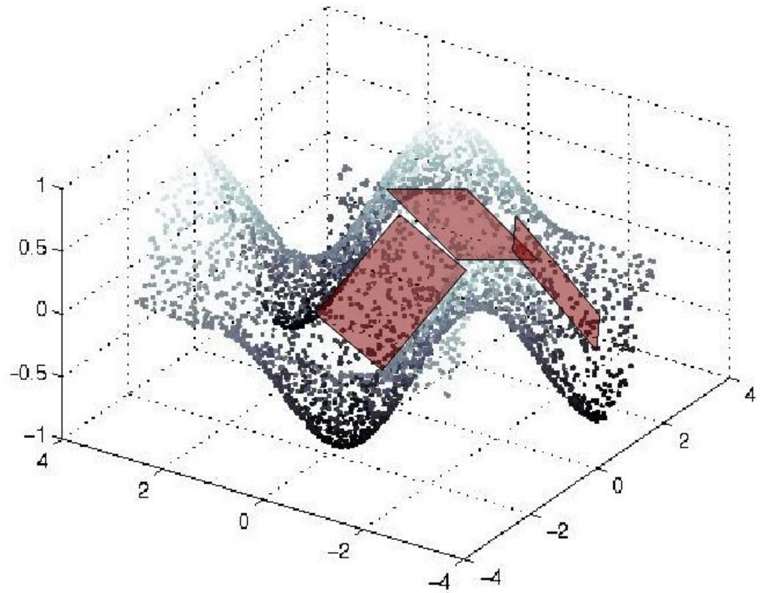



$W_2x + b$
$W_1x + b$
ReLU
$W_2x + b$
$W_1x + b$

Rectified Linear Unit

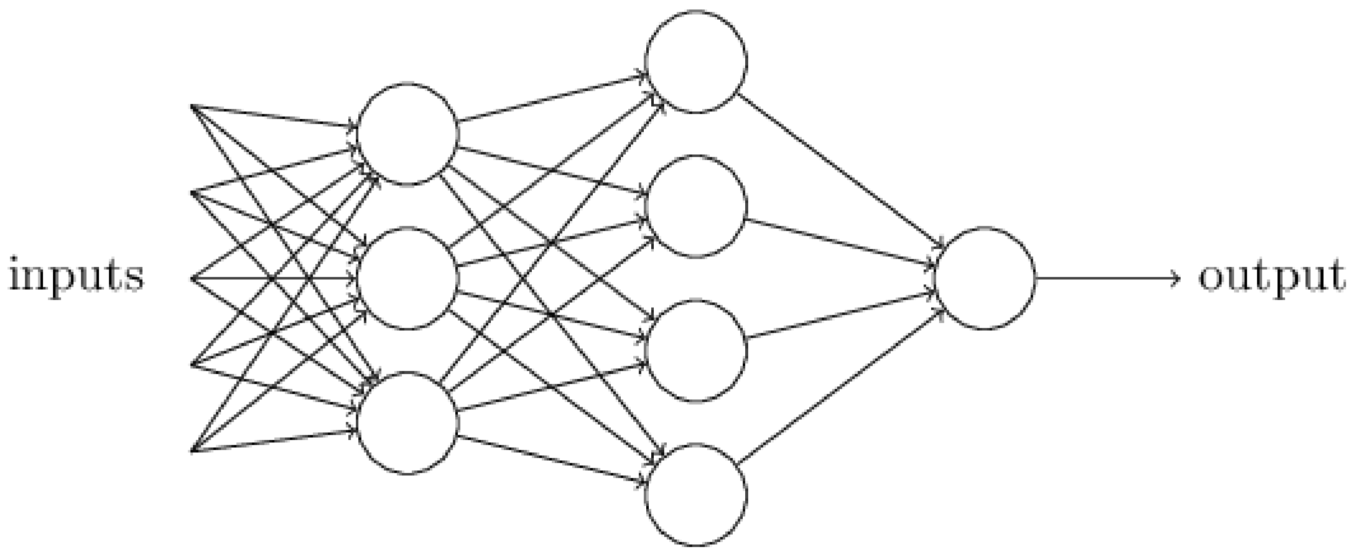
Question: What do ReLU layers accomplish?

Answer: Piece-wise linear tiling: mapping is locally linear.



Multi-layer perceptron (MLP)

- …is a ‘*fully connected*’ neural network with non-linear activation functions.



- ‘*Feed-forward*’ neural network

Nielson

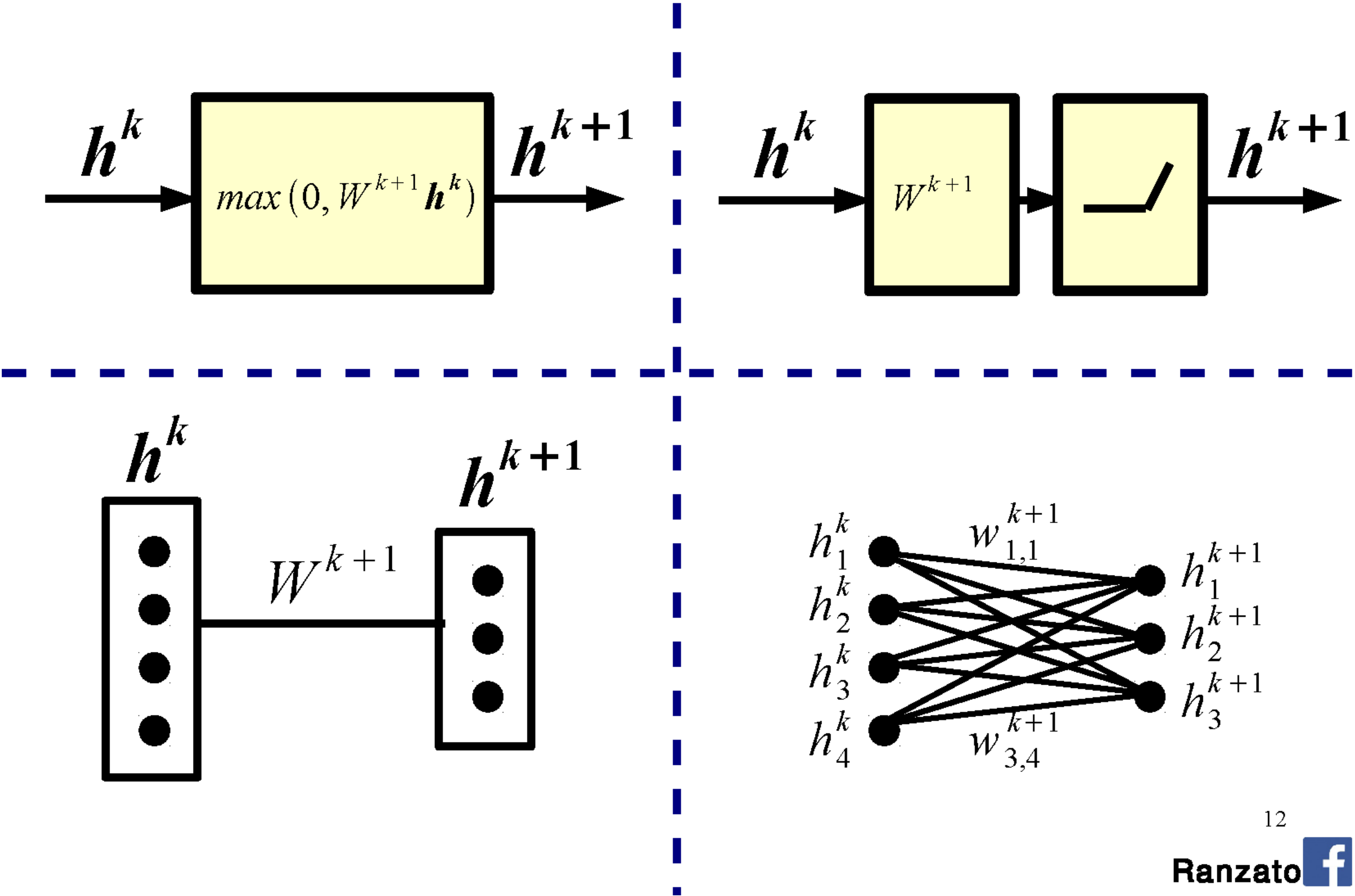
MLP

- Use is grounded in theory
  - Universal approximation theorem (Goodfellow 6.4.1)

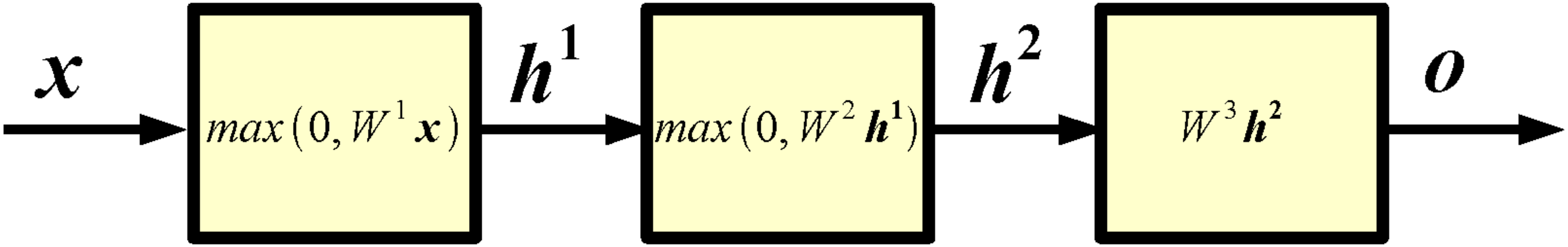
- Can represent a NAND circuit, from which any binary function can be built by compositions of NANDs

- With enough parameters, it can approximate any function.

Alternative Graphical Representation



Neural Networks: example



$\boldsymbol{x}$ input

$\boldsymbol{h}^1$ 1-st layer hidden units

$\boldsymbol{h}^2$ 2-nd layer hidden units

$\boldsymbol{o}$ output

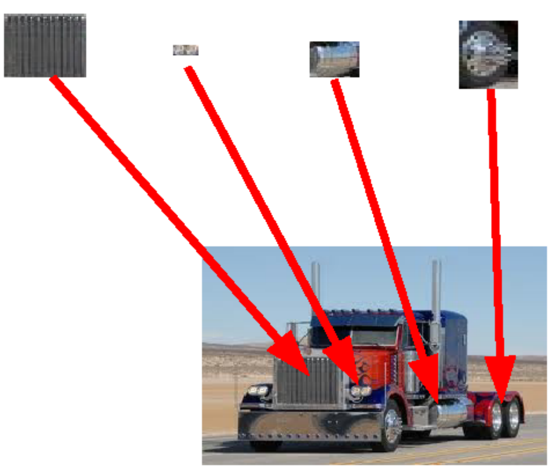
Example of a 2 hidden layer neural network (or 4 layer network, counting also input and output).

Interpretation

Question: Why do we need many layers?

Answer: When input has hierarchical structure, the use of a hierarchical architecture is potentially more efficient because intermediate computations can be re-used. DL architectures are efficient also because they use **distributed representations** which are shared across classes.

[0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 …] truck feature

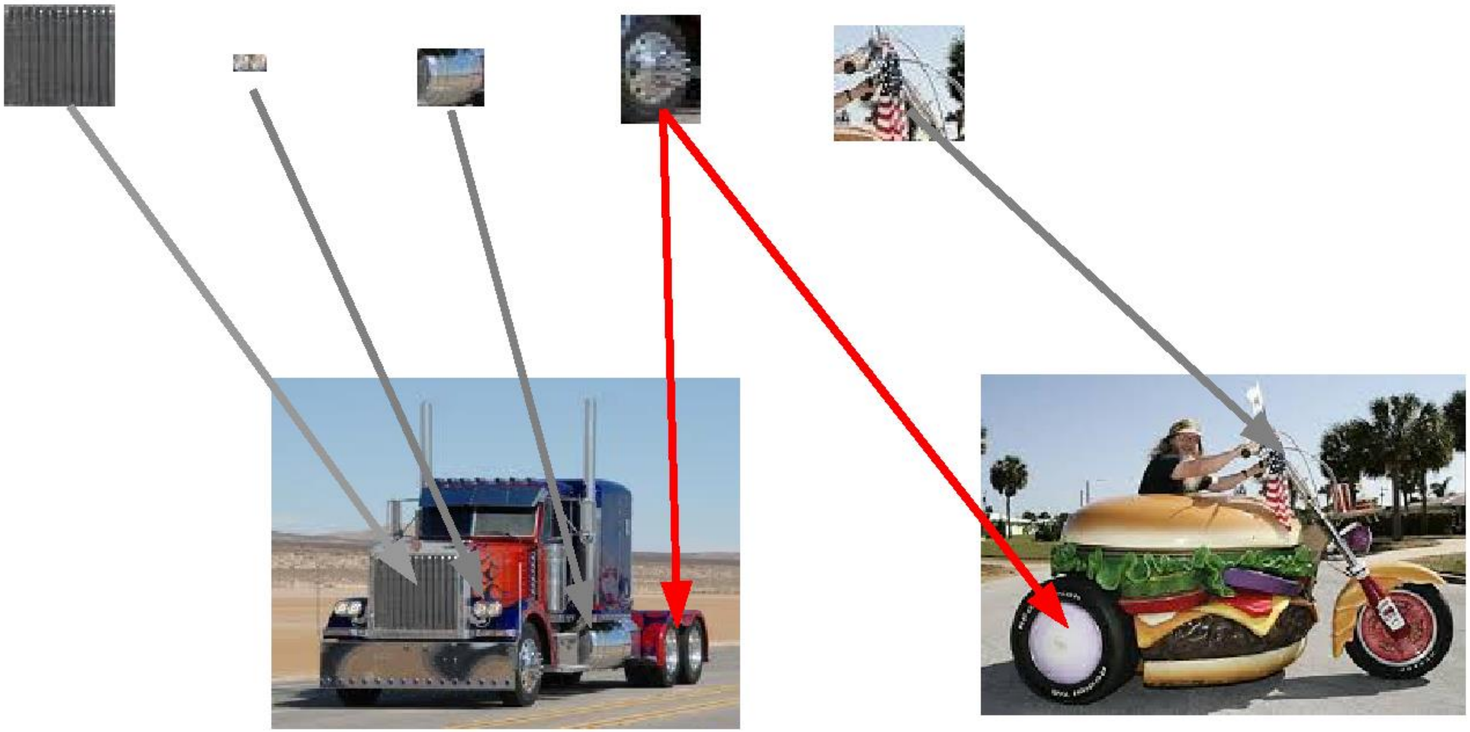


Exponentially more efficient than a 1-of-N representation (a la k-means)

Interpretation

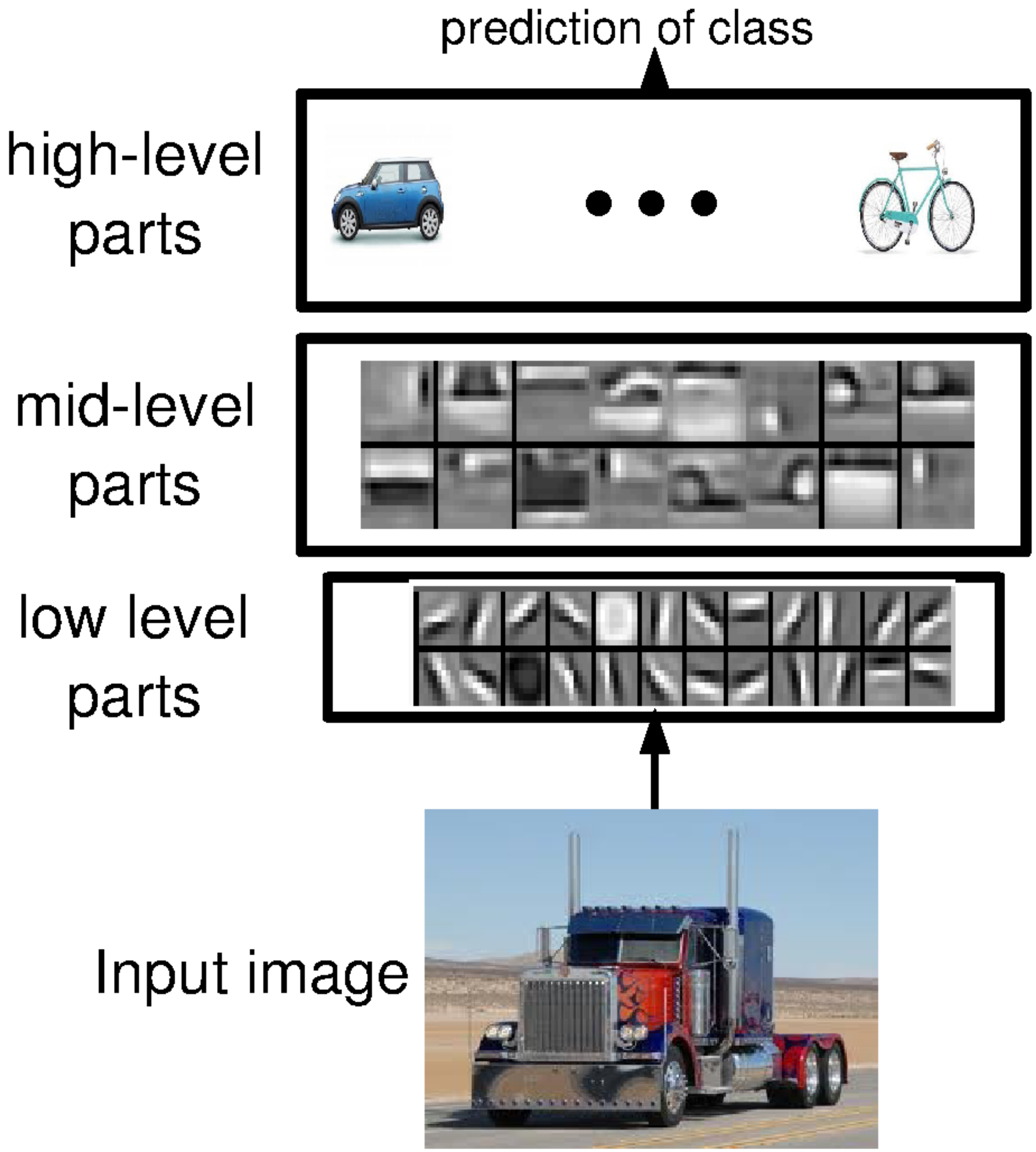
[1 1 0 0 0 1 0 **1** 0 0 0 0 1 1 0 1…] motorbike

[0 0 1 0 0 0 0 **1** 0 0 1 1 0 0 1 0 …] truck



Ranzato

Interpretation



- distributed representations
- feature sharing
- compositionality

Lee et al. “Convolutional DBN's ...” ICML 2009

Interpretation

Question: What does a hidden unit do?

Answer: It can be thought of as a classifier or feature detector.

Question: How many layers? How many hidden units?

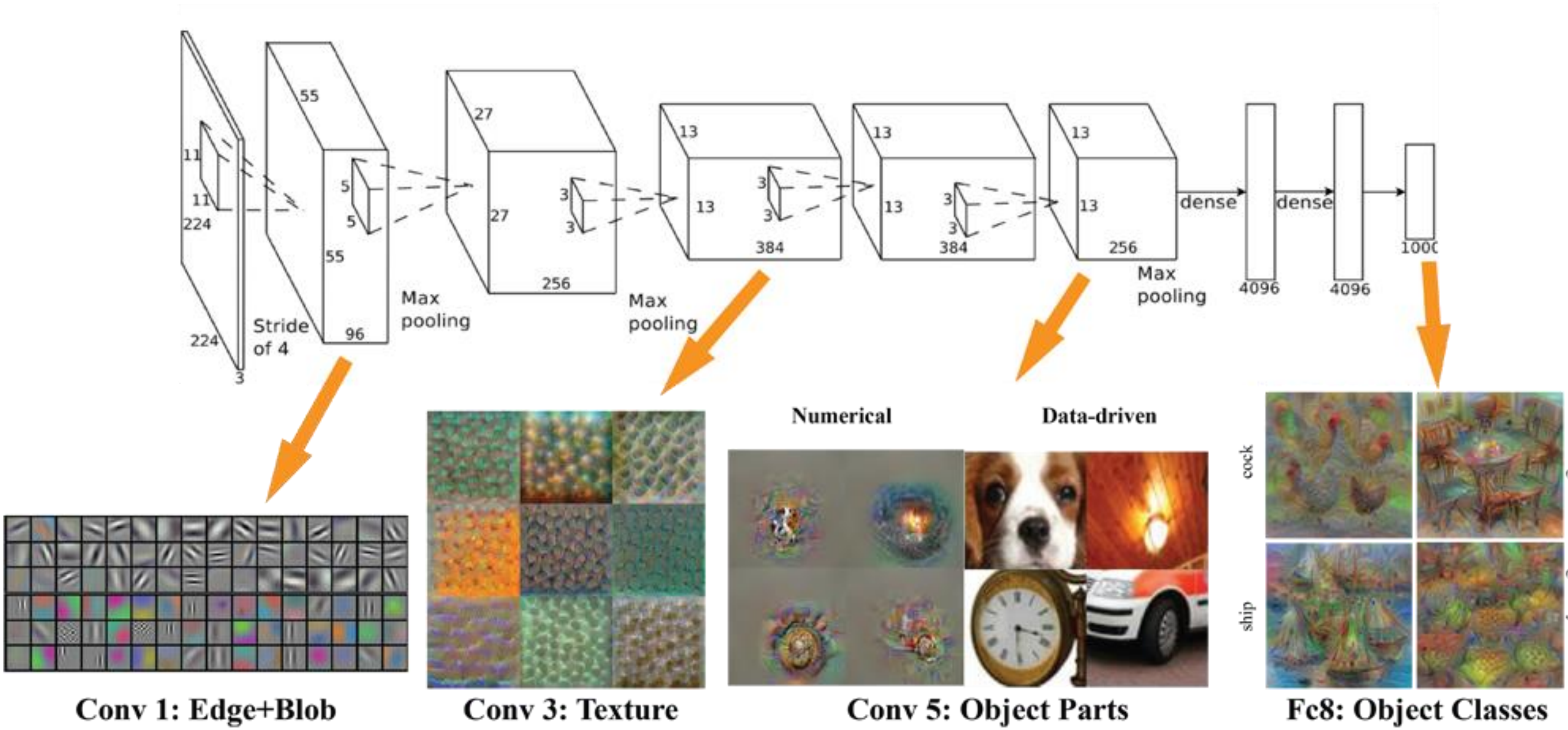
Answer: Cross-validation or hyper-parameter search methods are the answer. In general, the wider and the deeper the network the more complicated the mapping.

Question: How do I set the weight matrices?

Answer: Weight matrices and biases are learned.
First, we need to define a measure of quality of the current mapping.
Then, we need to define a procedure to adjust the parameters.

Project 6 out today

- Good luck finishing project 5!



Wei et al.