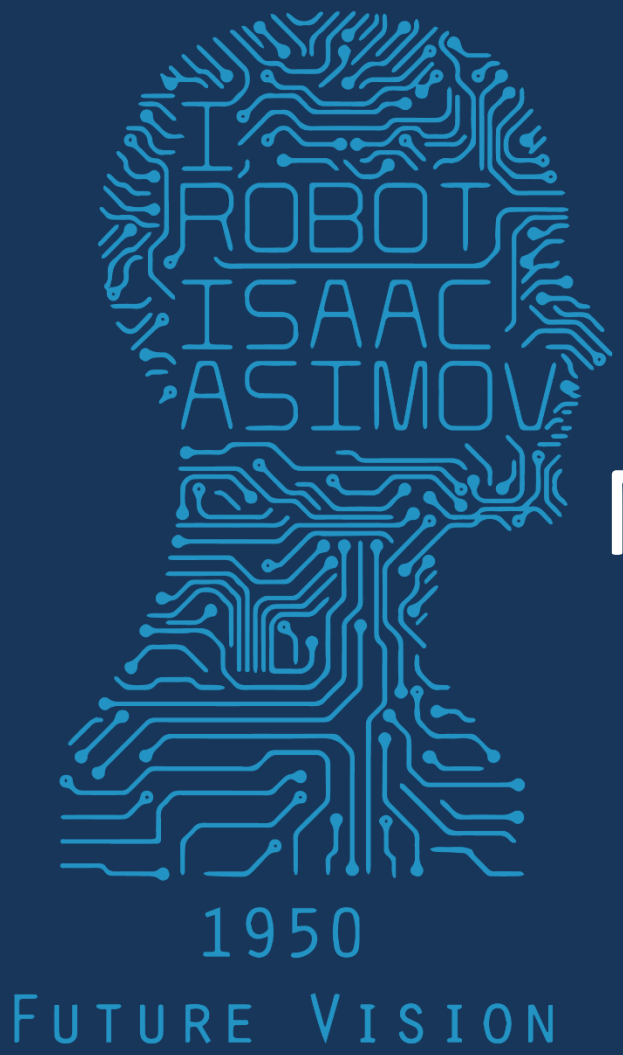
Advanced Computer Vision



Neural Networks

!!! Warning !!!

Learning similar images is always painful…

…even if the concepts behind the images 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)

The limits of learning?

So far…

For year 2012

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

Smart human brains used intuition and understanding of how we think vision works, and it's pretty good.

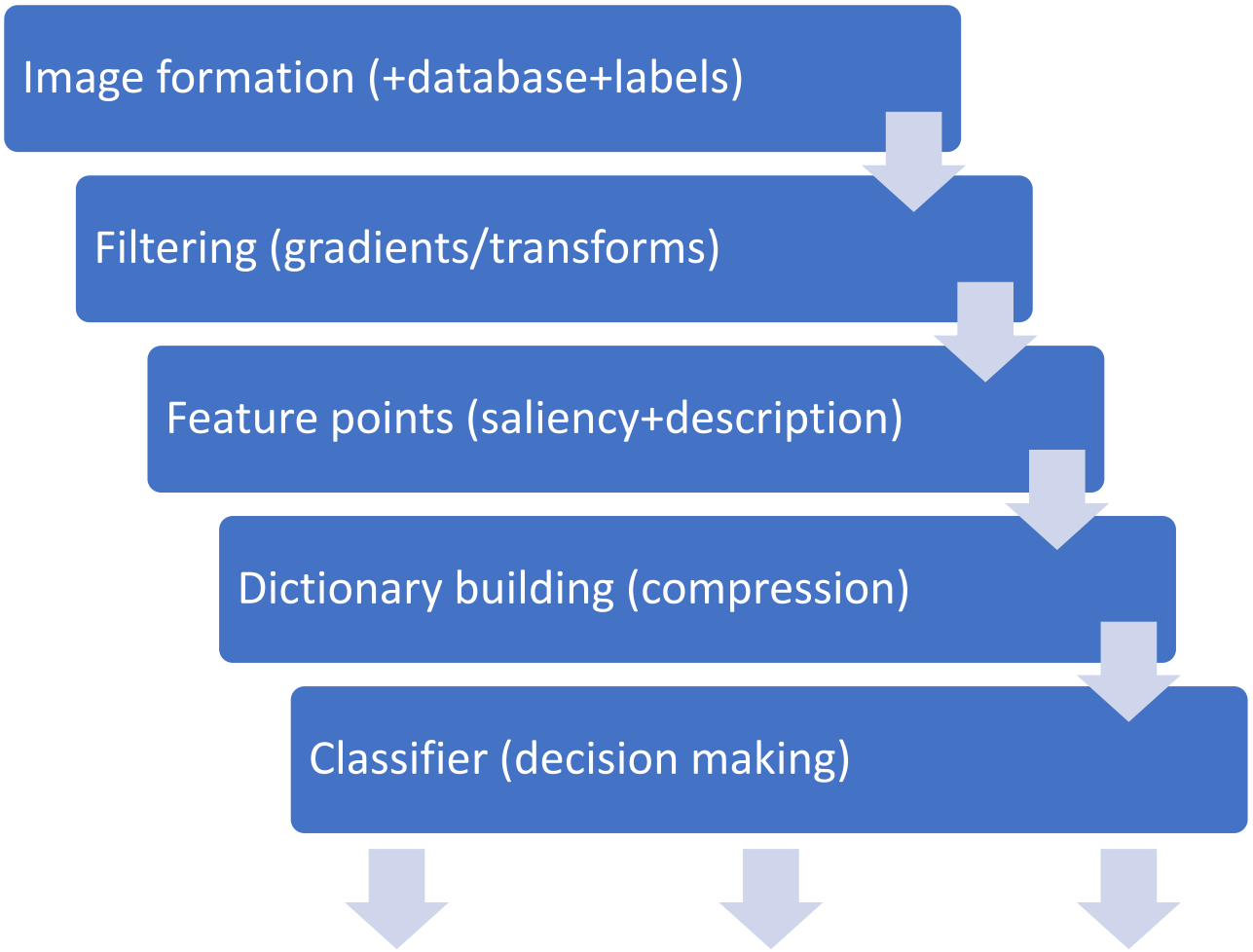
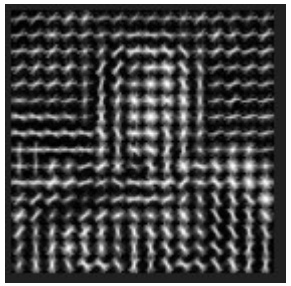


Image formation (+database+labels)
Filtering (gradients/transforms)
Feature points (saliency+description)
Dictionary building (compression)
Classifier (decision making)
Captured+manual.
Hand designed.
Hand designed.
Hand designed.
Learned.
Recognition:
Classification
Object Detection
Segmentation

Well, what do we have?

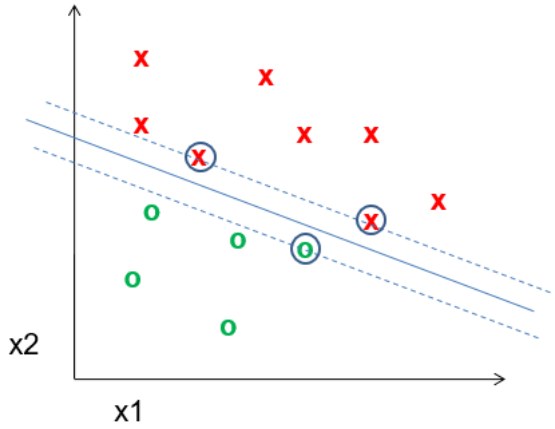
Best performing visions systems have commonality:

- 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%

Problems:

- Lossy features
- Lossy quantization
- Imperfect classifier

But it's still not that good…

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

How to solve?

- *Features: More principled modeling?*
  We *know* why the world looks (it's physics!);
  Let's build better physically-meaningful models.
- *Quantization: More data and more compute?*
  It's just an interpolation problem; let's represent the space with less approximation.
- *Classifier: …*

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

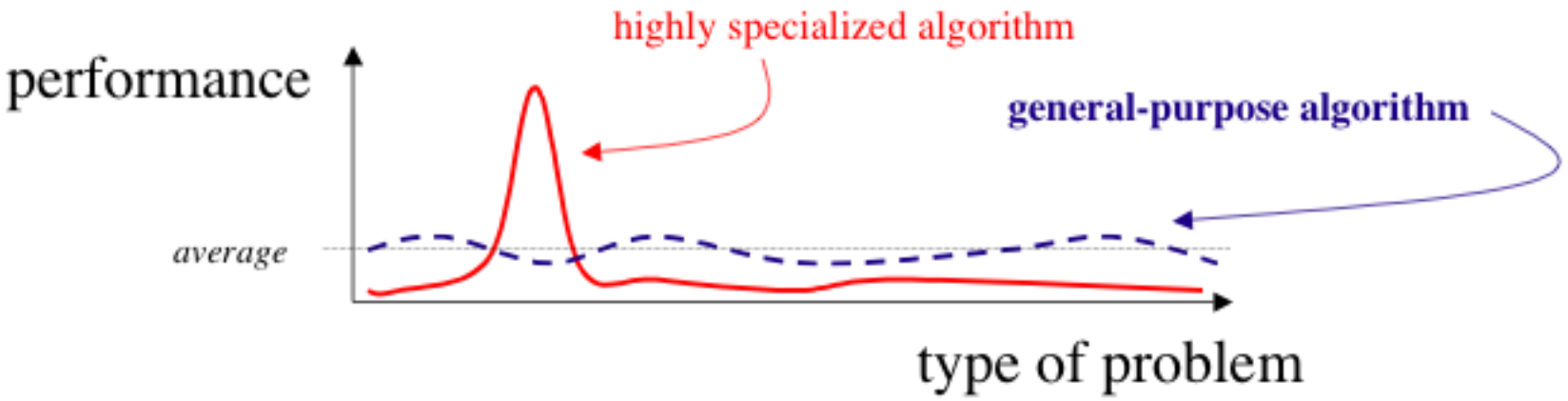
Hume (c.1739):

"'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.

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?

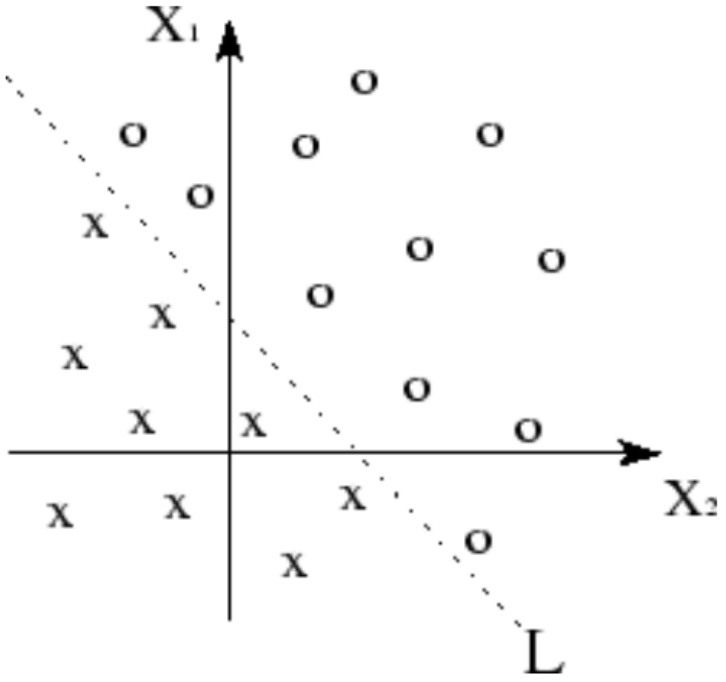
- PASCAL VOC = ~75%
- ImageNet = ~75%; human performance = ~95%

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.



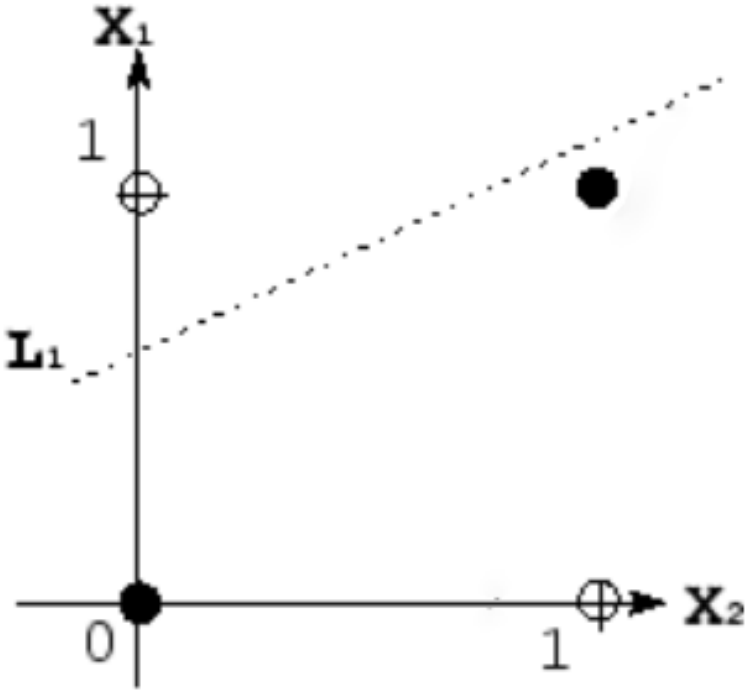
Linearly separable data
+ linear classifer = good.

Non-linear spaces - separability

- Linear functions are insufficient on their own.

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

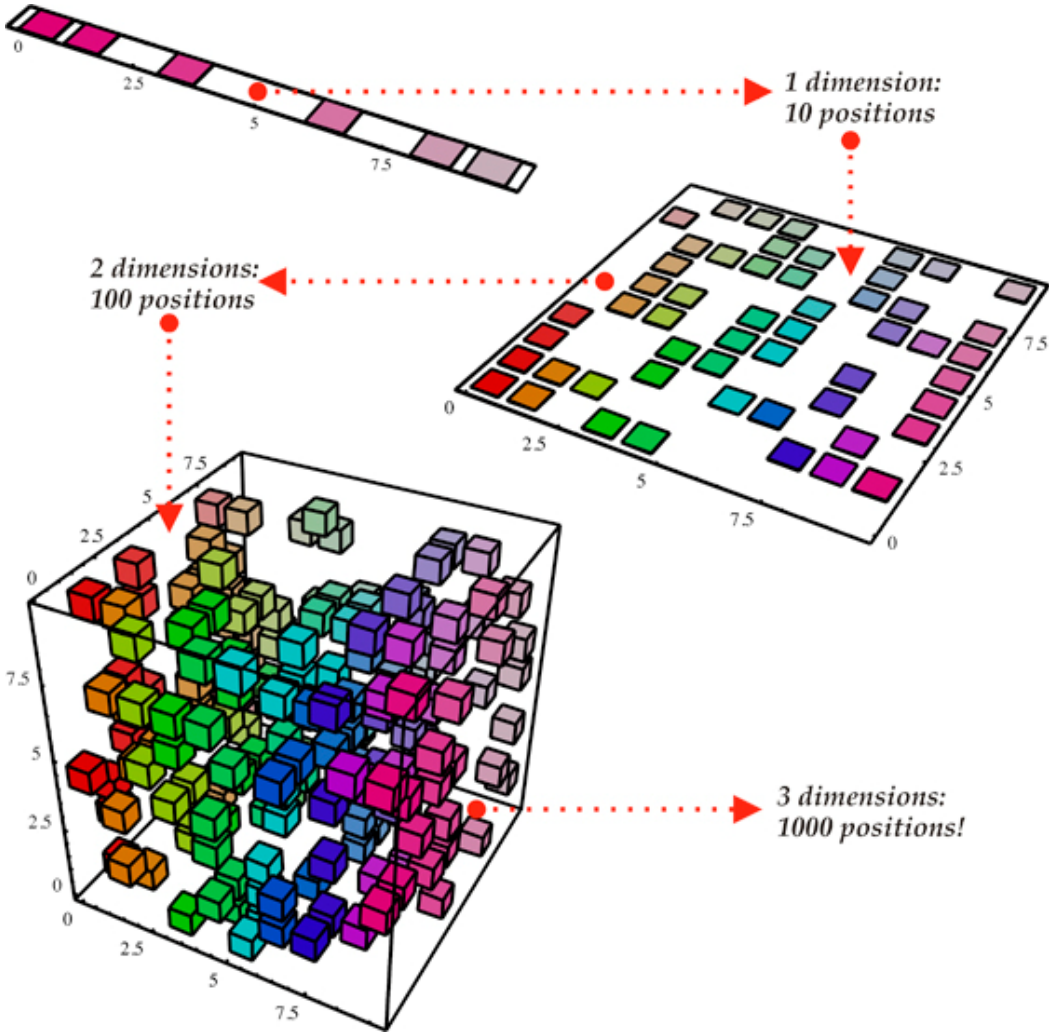
$Y = X_1 \oplus X_2$



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

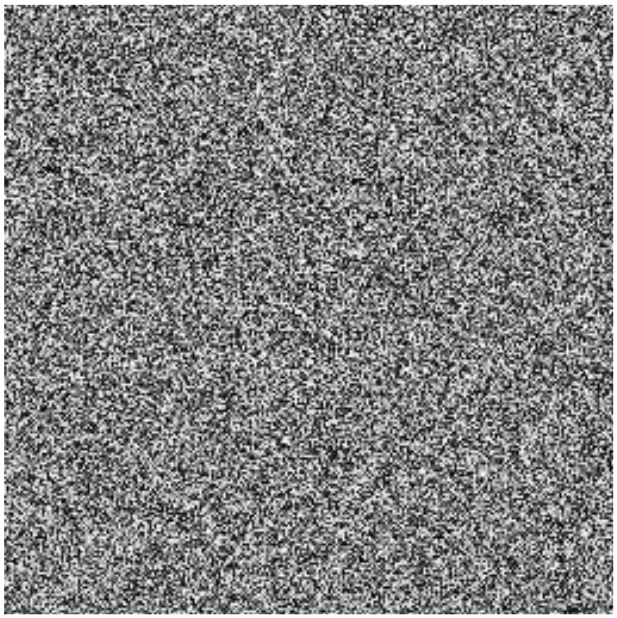


Curse of Dimensionality

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

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

@ 8bit = 256 values ^ 65,536



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

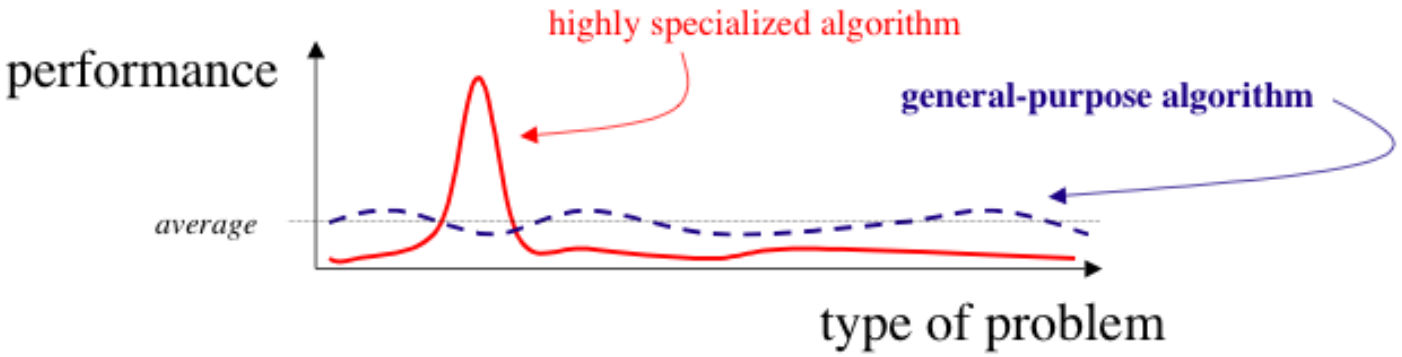
- At the extreme: Take k-NN classifier.
- The number of regions cannot be more than the number of examples.

-> No way to generalize beyond examples

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

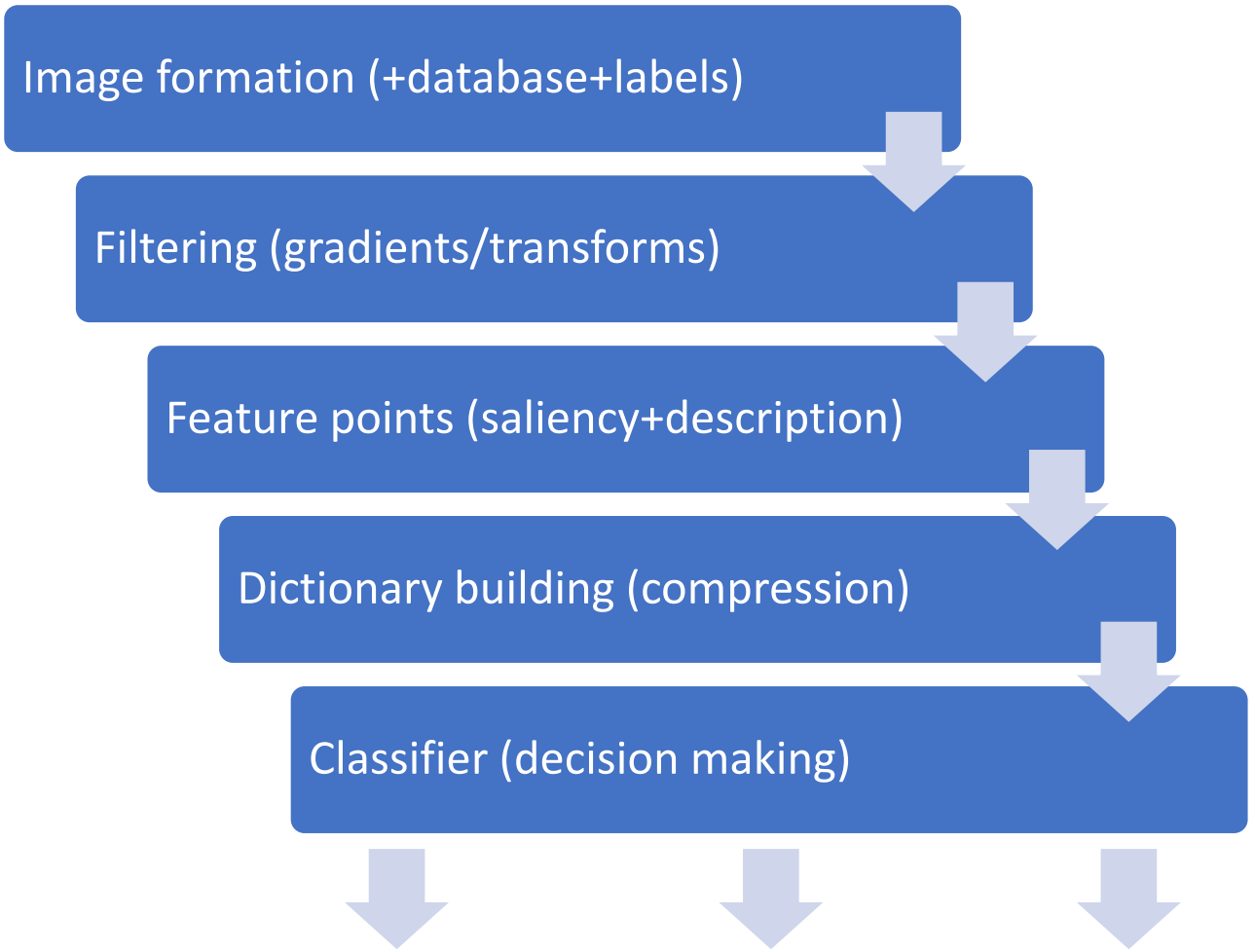
More specialization?

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

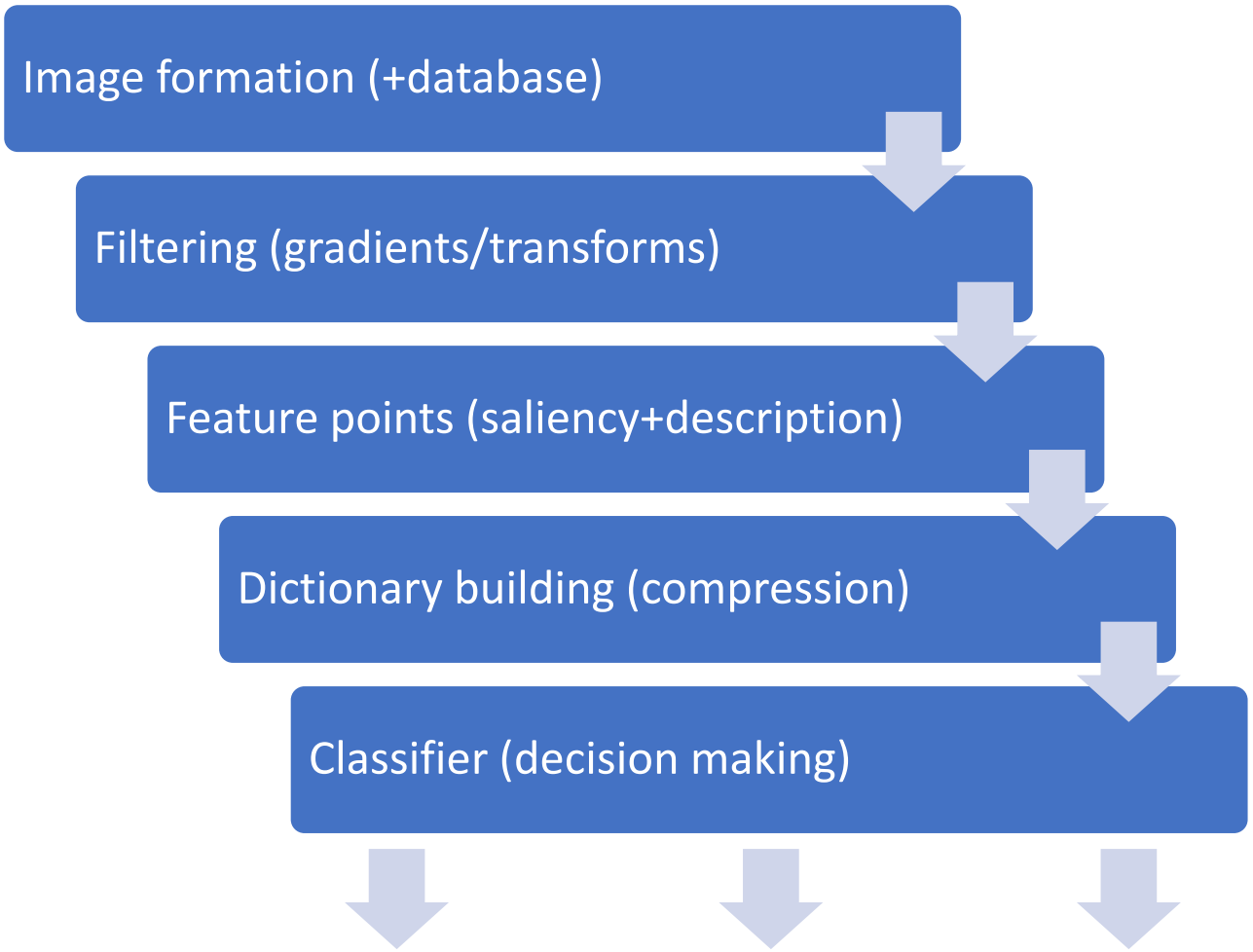


Is there a way to make our system better suited to the problem?

Wouldn’t it be great if we could…



Well if we can do that, then what about…



Captured+no labels.

Learned.
(space specified a bit)

Learned.

Learned.

Learned.

Unsupervised

End to end learning!

Recognition: Classification Object Detection Segmentation

Goals

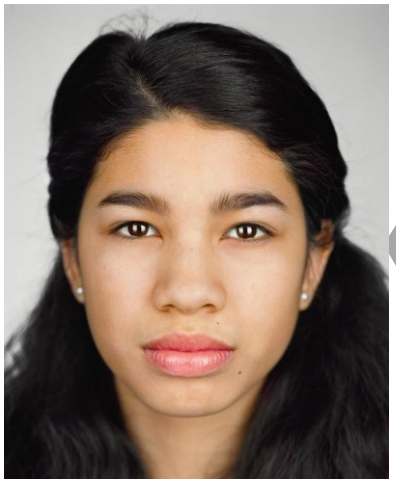
Build a classifier which is more powerful
at representing complex functions
and more suited to the learning problem.

What does this mean?

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

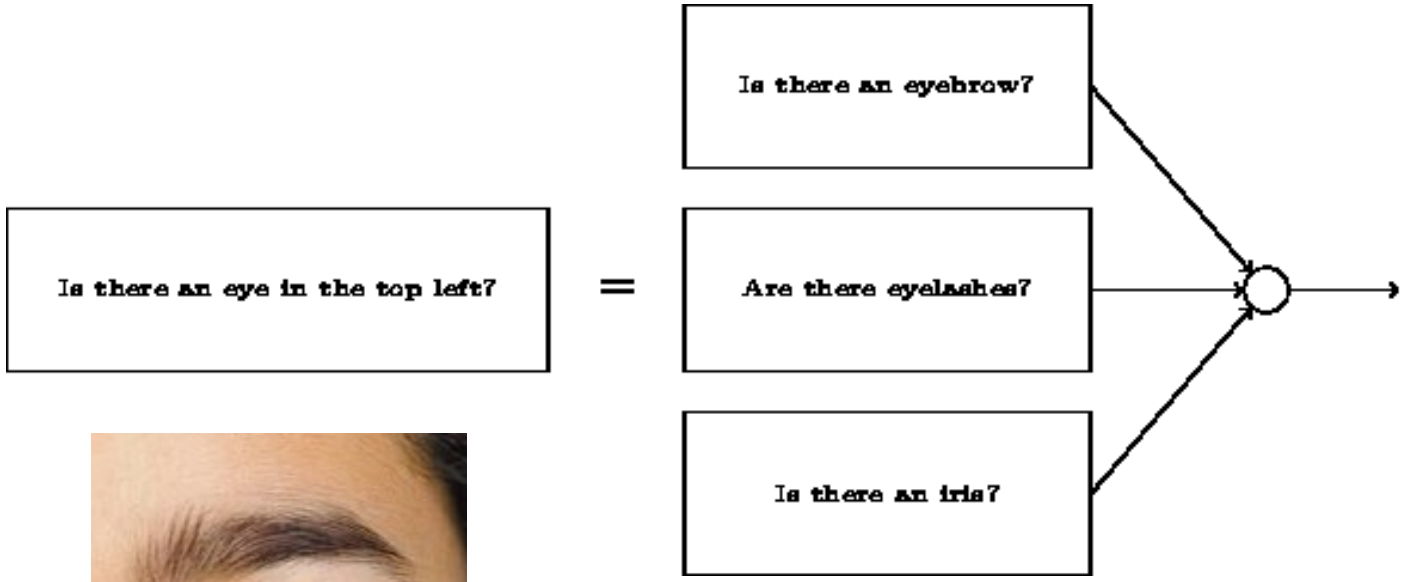
Dependencies between regions in feature space
= factor composition

Example



Nielsen, National Geographic

Example



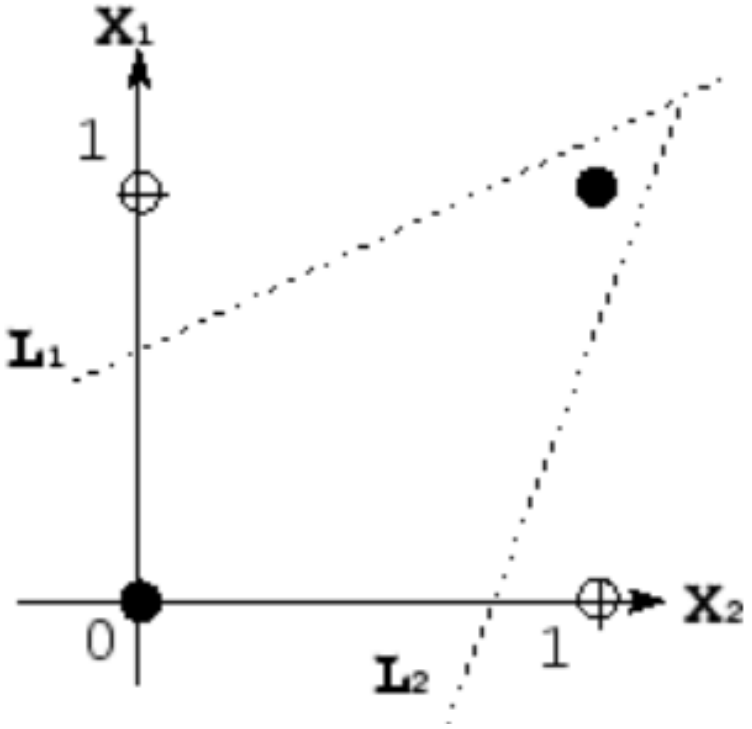
Nielsen, National Geographic

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

$Y = X_1 \oplus X_2$



Goals

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

What does this mean?

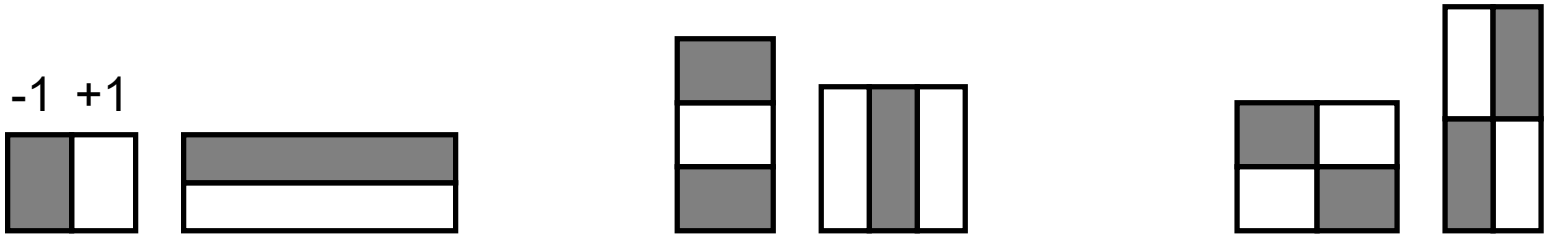
1. Assume that the *underlying data generating function* relies on a composition of factors in a hierarchy.

2. Learn a feature representation specific to the dataset.

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

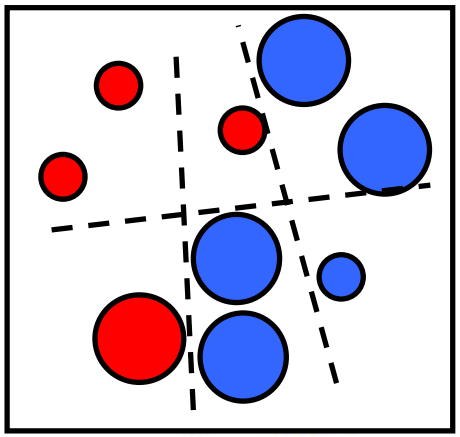
Reminder: Viola Jones Face Detector

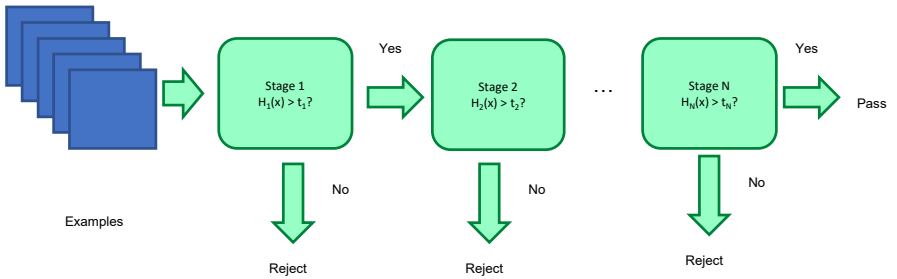
Combine *thousands* of 'weak classifiers'



Two-rectangle features | Three-rectangle features | Etc.

Learn how to combine in cascade with boosting





Viola Jones

Image formation
(+database+labels)

Captured+manual.

Features
(saliency+description)

Specified space, but selected automatically.

Classifier

(decision making)

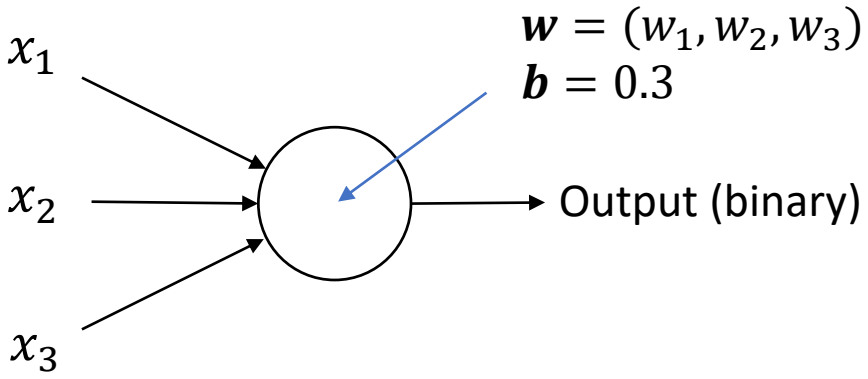
Learned combination.

Recognition: Object Detection

Neural Networks

Neural Networks

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



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

$$w \cdot x \equiv \sum_j w_j x_j$$

Binary classifying an image

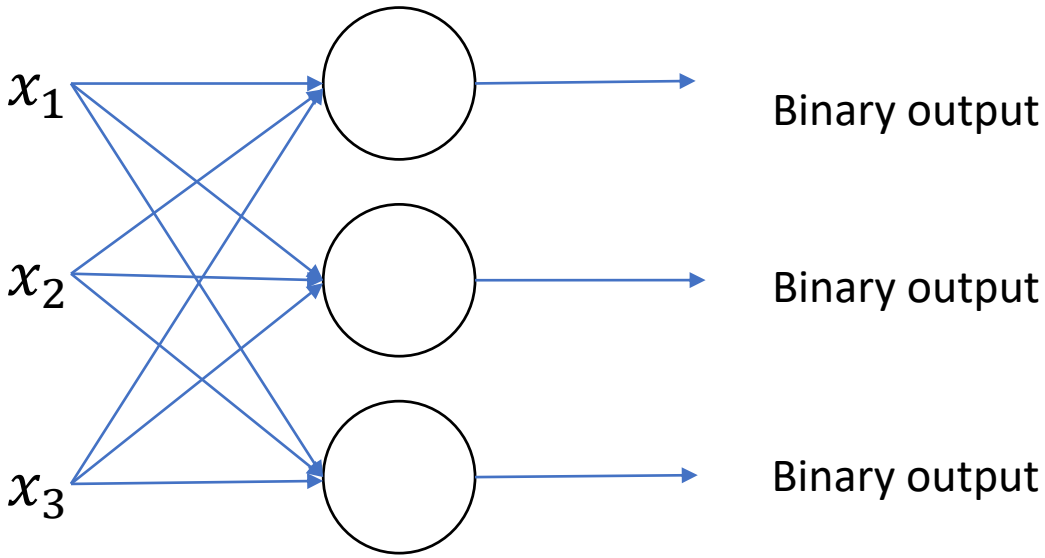
- Each pixel of the image would be an input.
- So, for a 28 x 28 image, we vectorize.
- **x** = 1 x 784

- **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

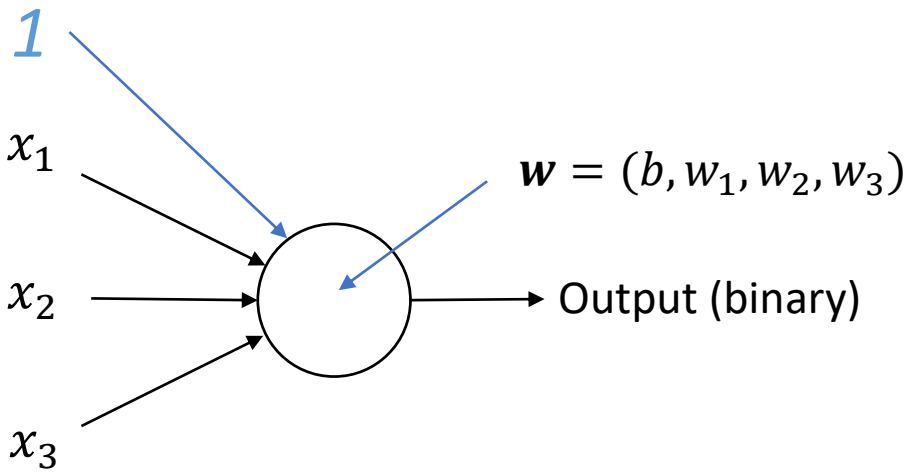
- Each pixel of the image would be an input.
- So, for a 28 x 28 image, we vectorize.
- **x** = 1 x 784

- **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

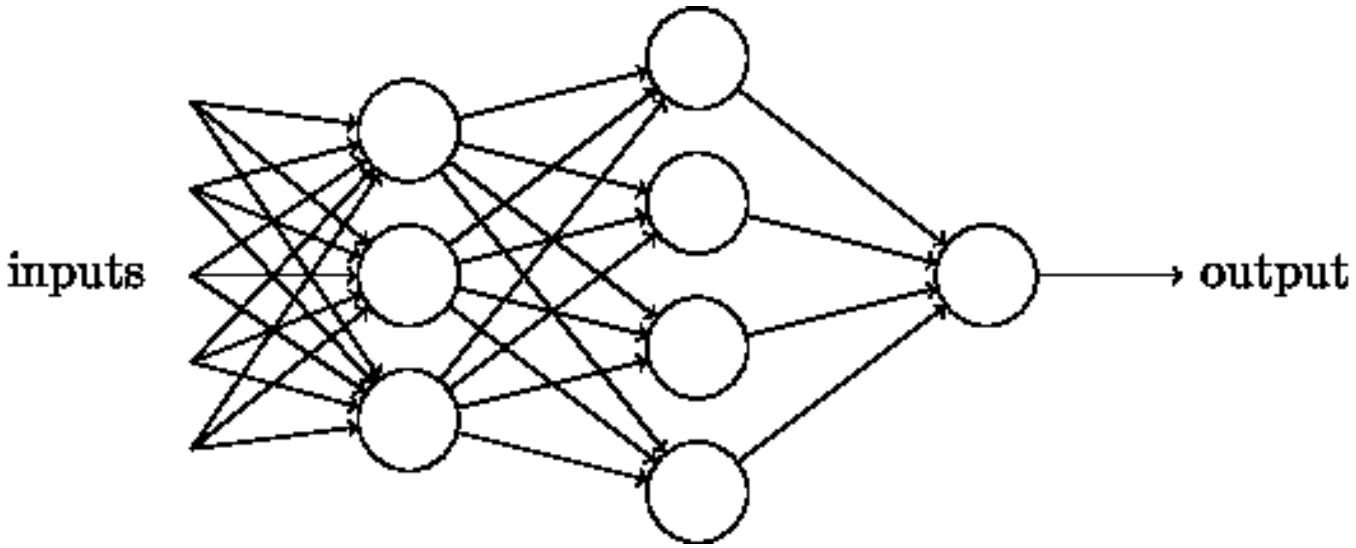
Bias convenience

- 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



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

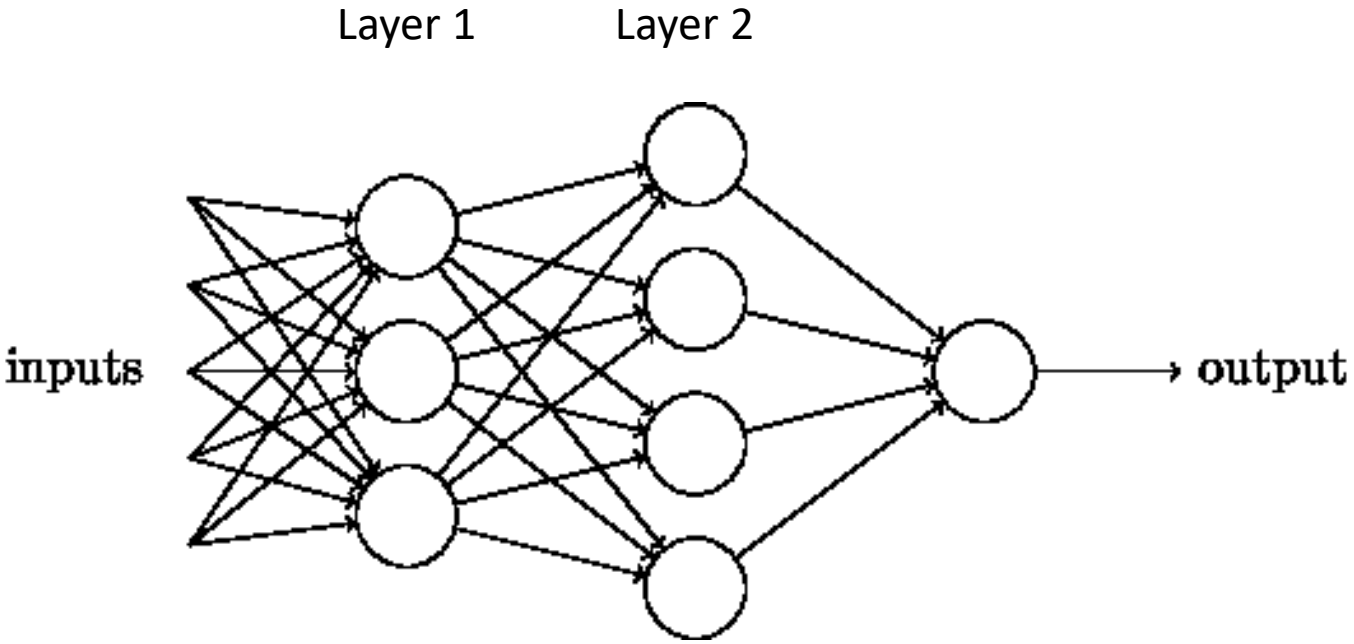
Composition



Attempt to represent complex functions as compositions of smaller functions.

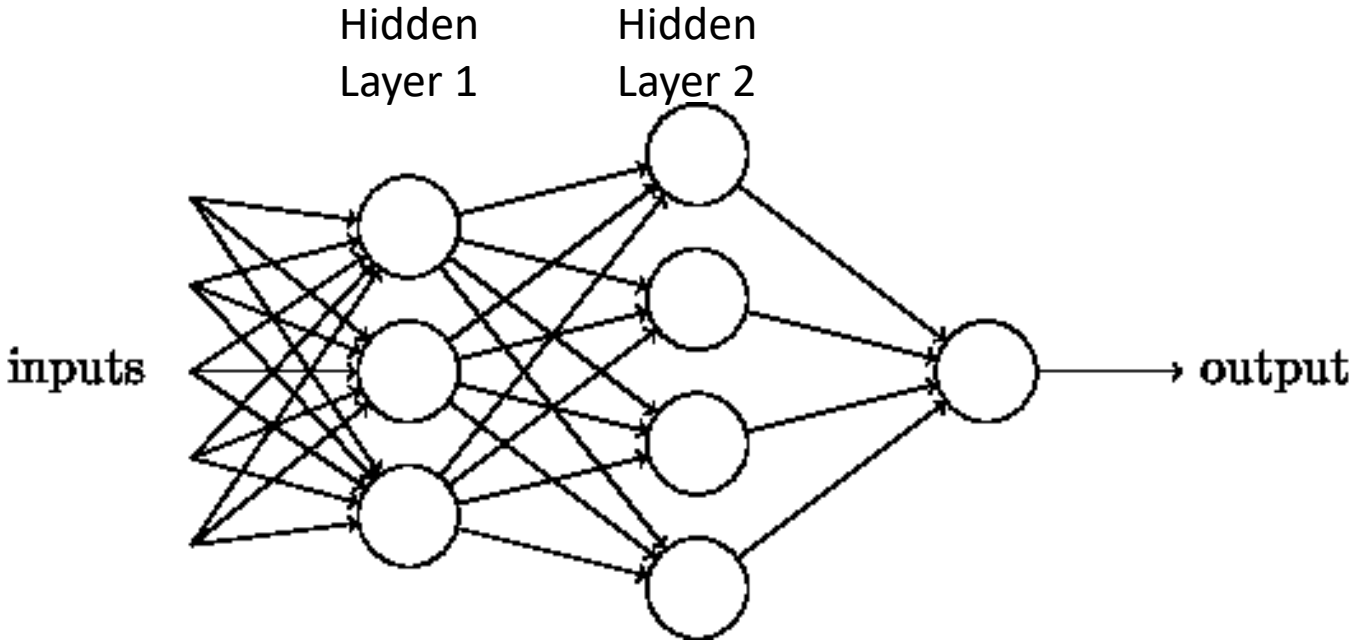
Outputs from one perception are fed into inputs of another perceptron.

Composition



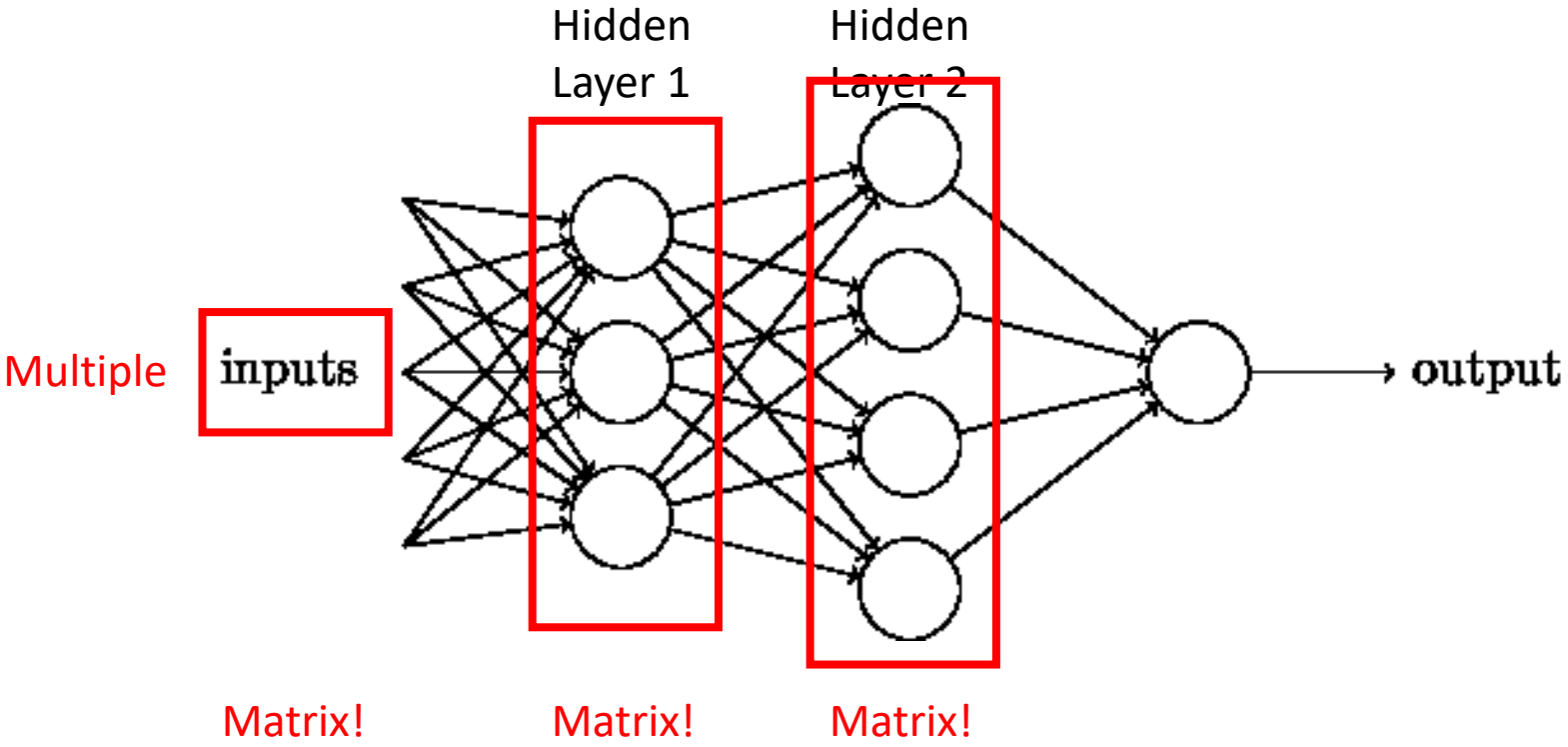
Sets of layers and the connections (weights) between them define the *network architecture.*

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.

Composition



It's all just matrix multiplication!

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

Problem 1 with all linear functions

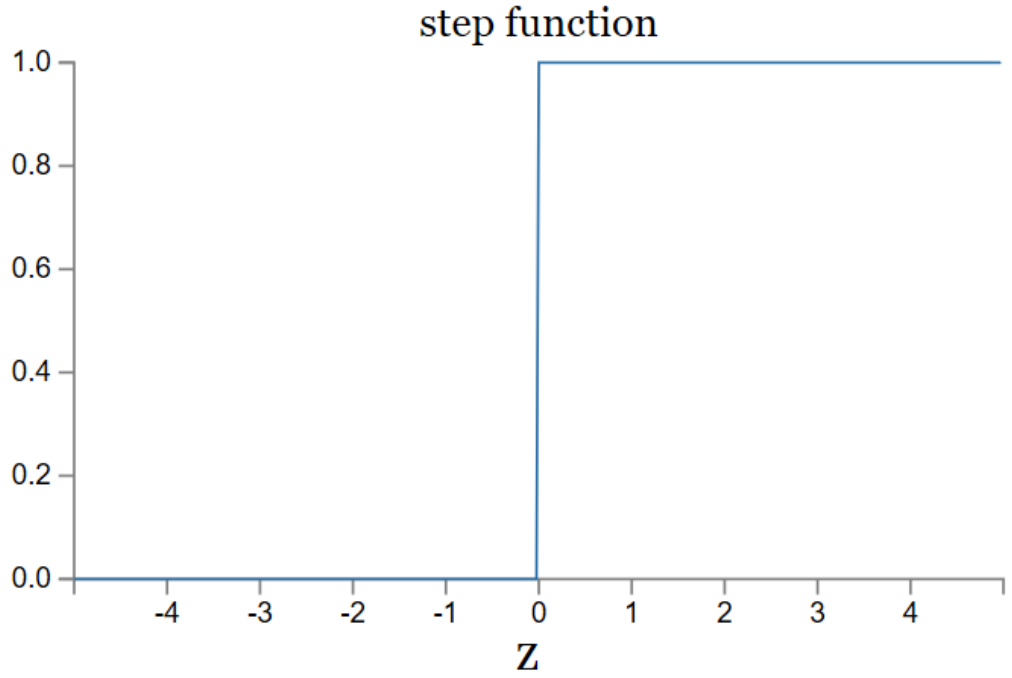
- We have formed chains of linear functions.
- We know that linear functions can be reduced
  - g = f(h(x))

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
  = problem for composition of functions

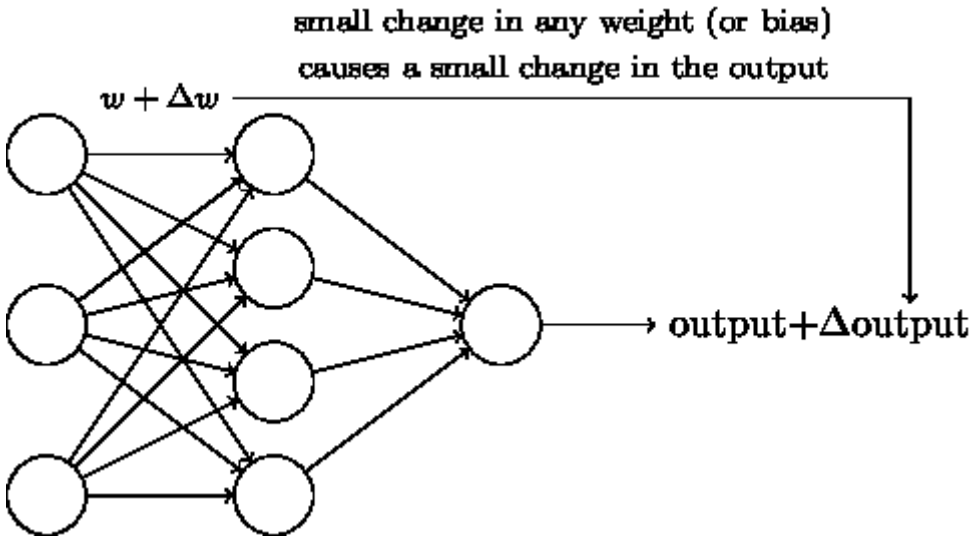
Activation function



Problem 2 with all linear functions

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

- We want:

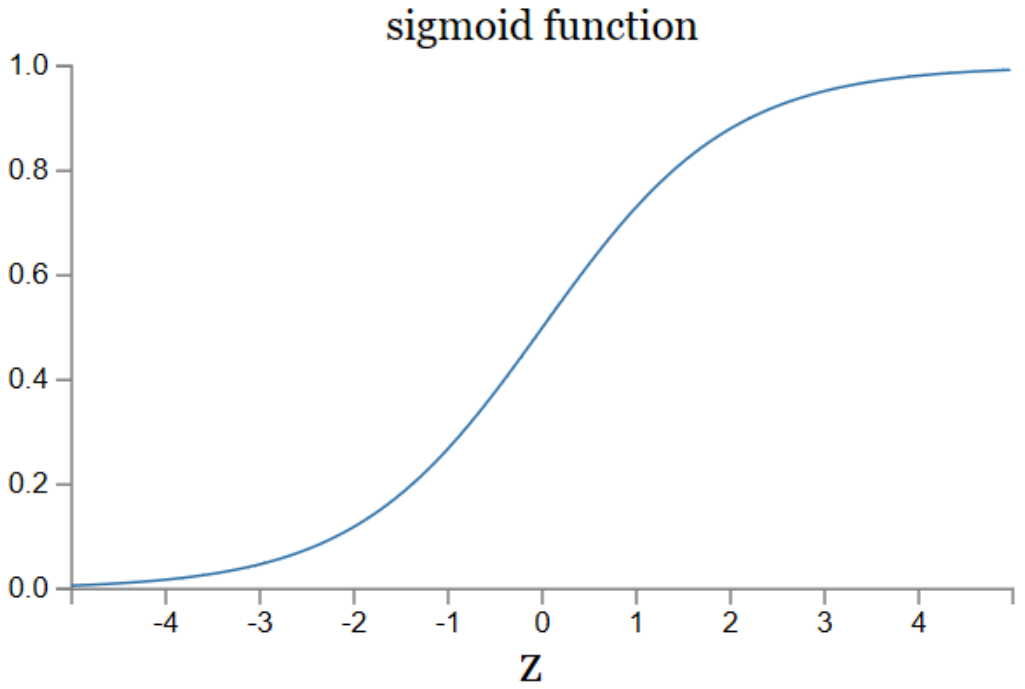


Let's introduce non-linearities

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

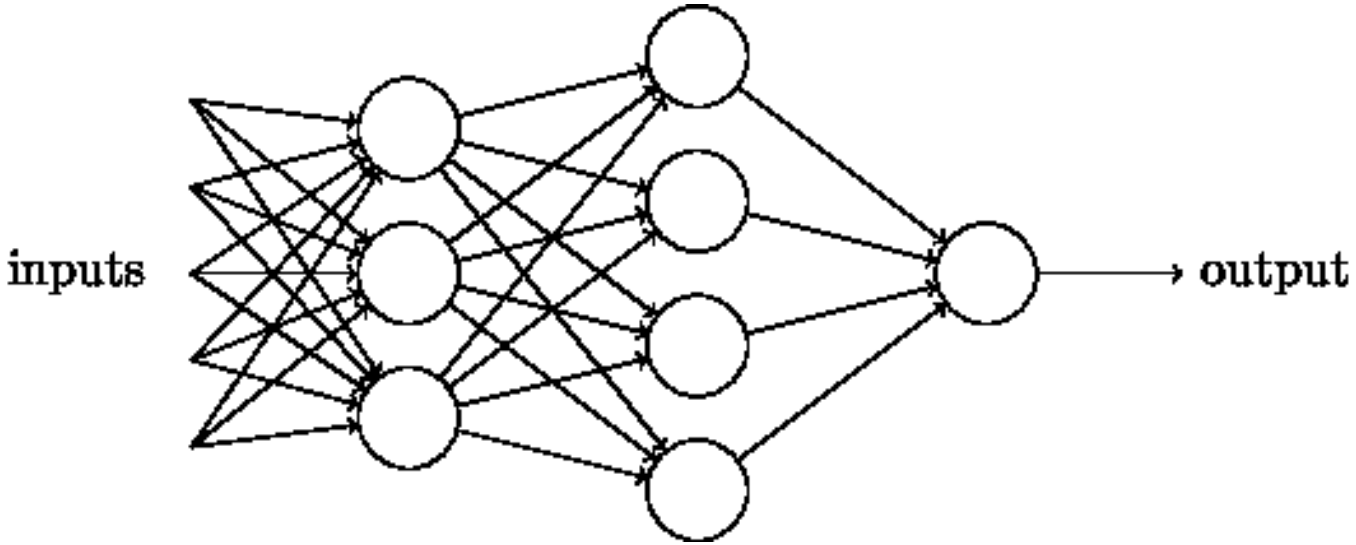
$$\sigma(w \cdot x + b)$$

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



Multi-layer perceptron (MLP)

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



- '*Feed-forward*' neural network

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 (next lecture).

Supervised Learning

$\{(\boldsymbol{x}^i, y^i), i = 1 \ldots P\}$ 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.

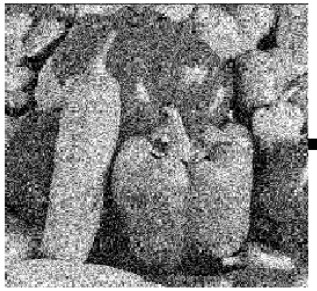
Supervised Learning: Examples

Classification

"dog"

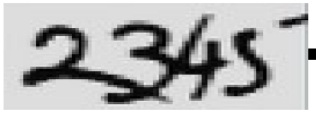
classification

Denoising



regression

OCR

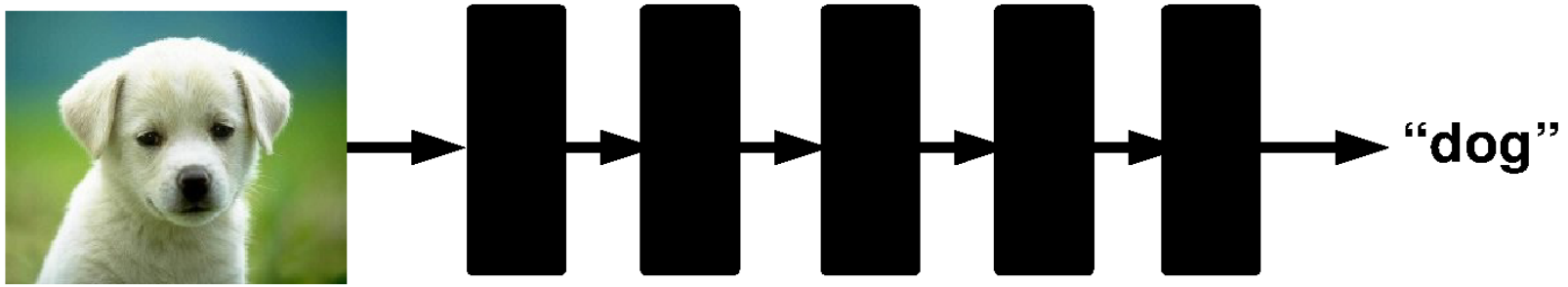


"2 3 4 5"

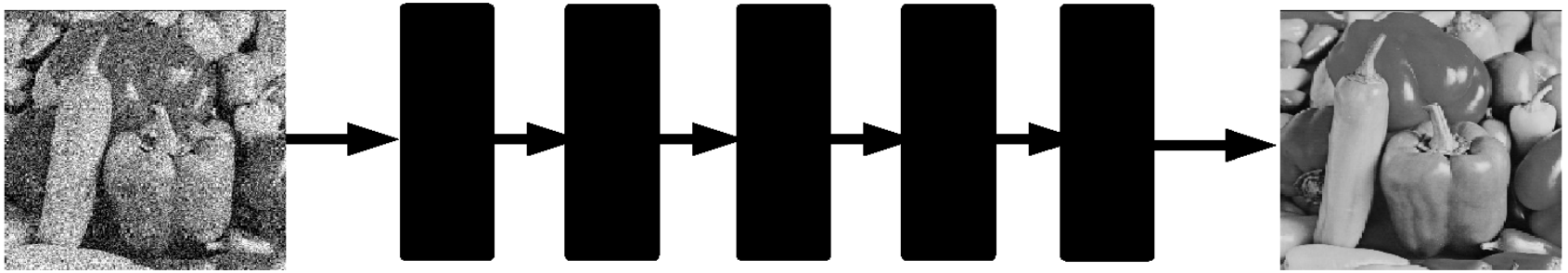
structured prediction

Supervised Deep Learning

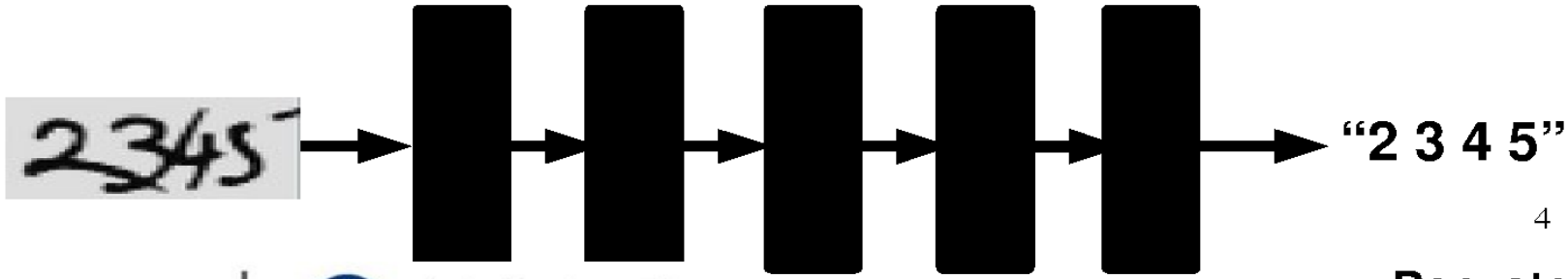
Classification



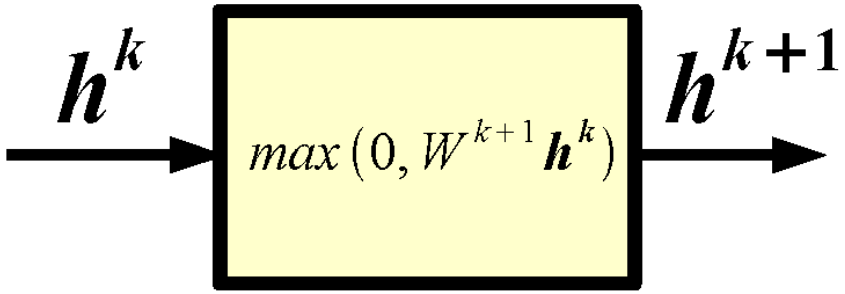
Denoising

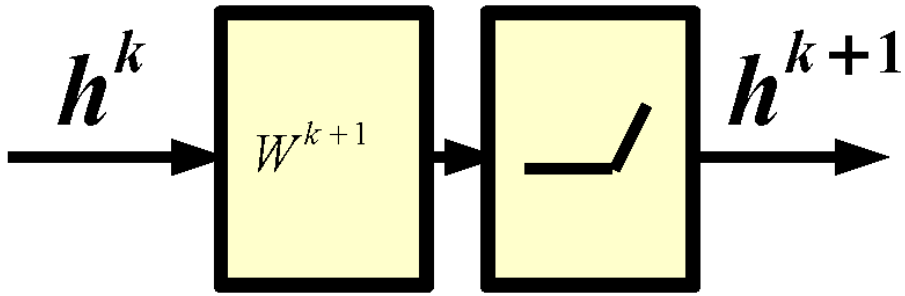


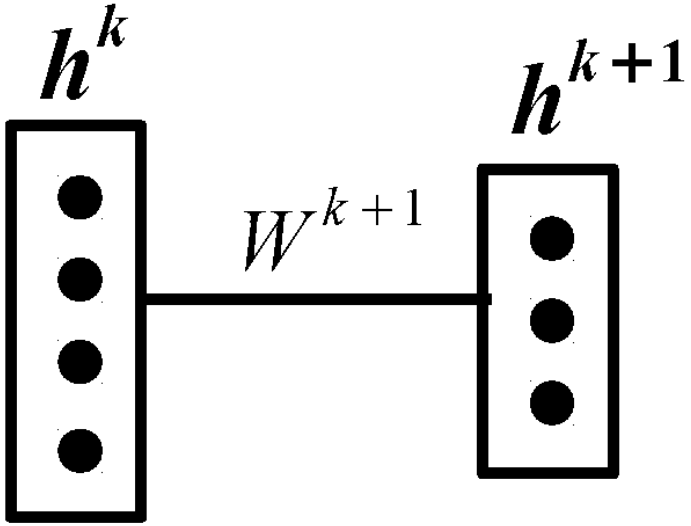
OCR

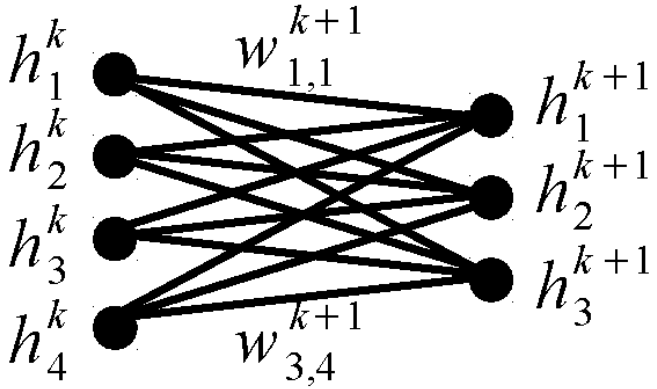


Alternative Graphical Representation

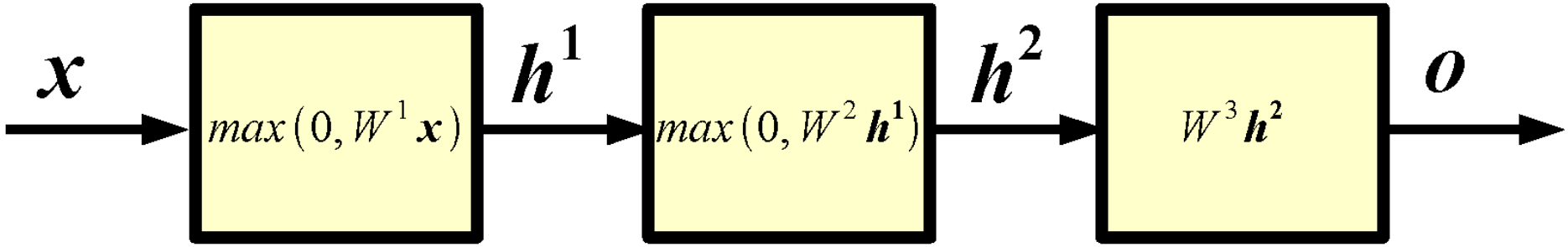








Neural Networks: example



x input

h^1 1-st layer hidden units

h^2 2-nd layer hidden units

o output

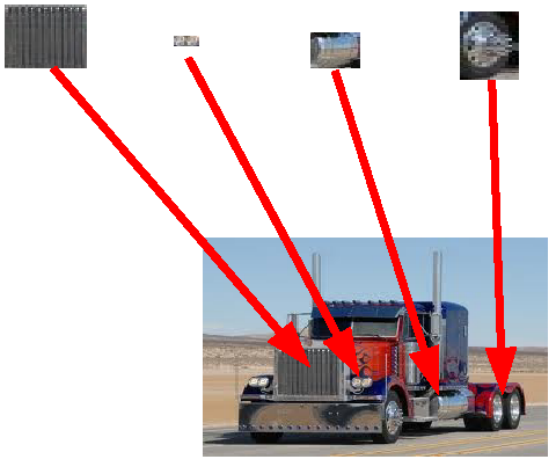
Example of a 2 hidden layer neural network (or 4 layer network, counting also input and output).

Why do we need many layers?

- A hierarchical structure is potentially more efficient because we can reuse intermediate computations.

- Different representations can be distributed across classes.

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



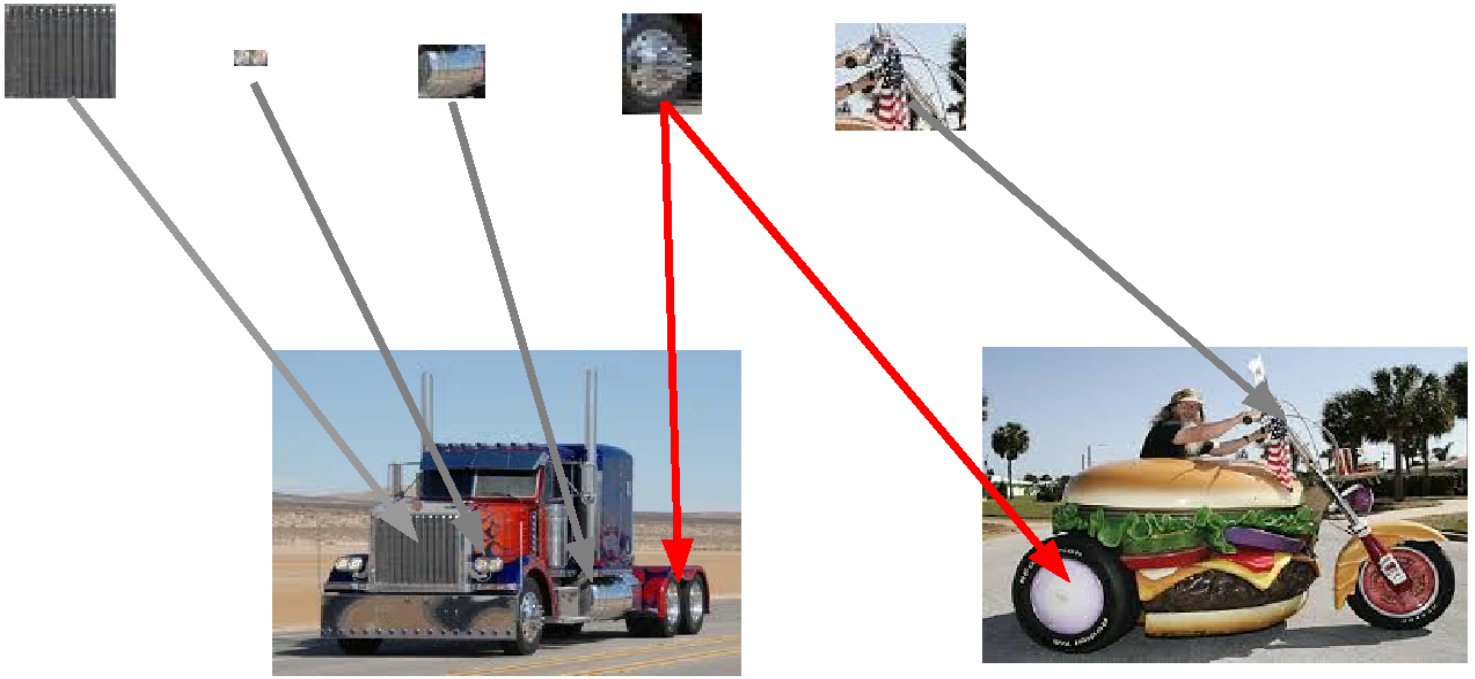
Exponentially more efficient than a 1-of-N representation (a la k-means)

Ranzato

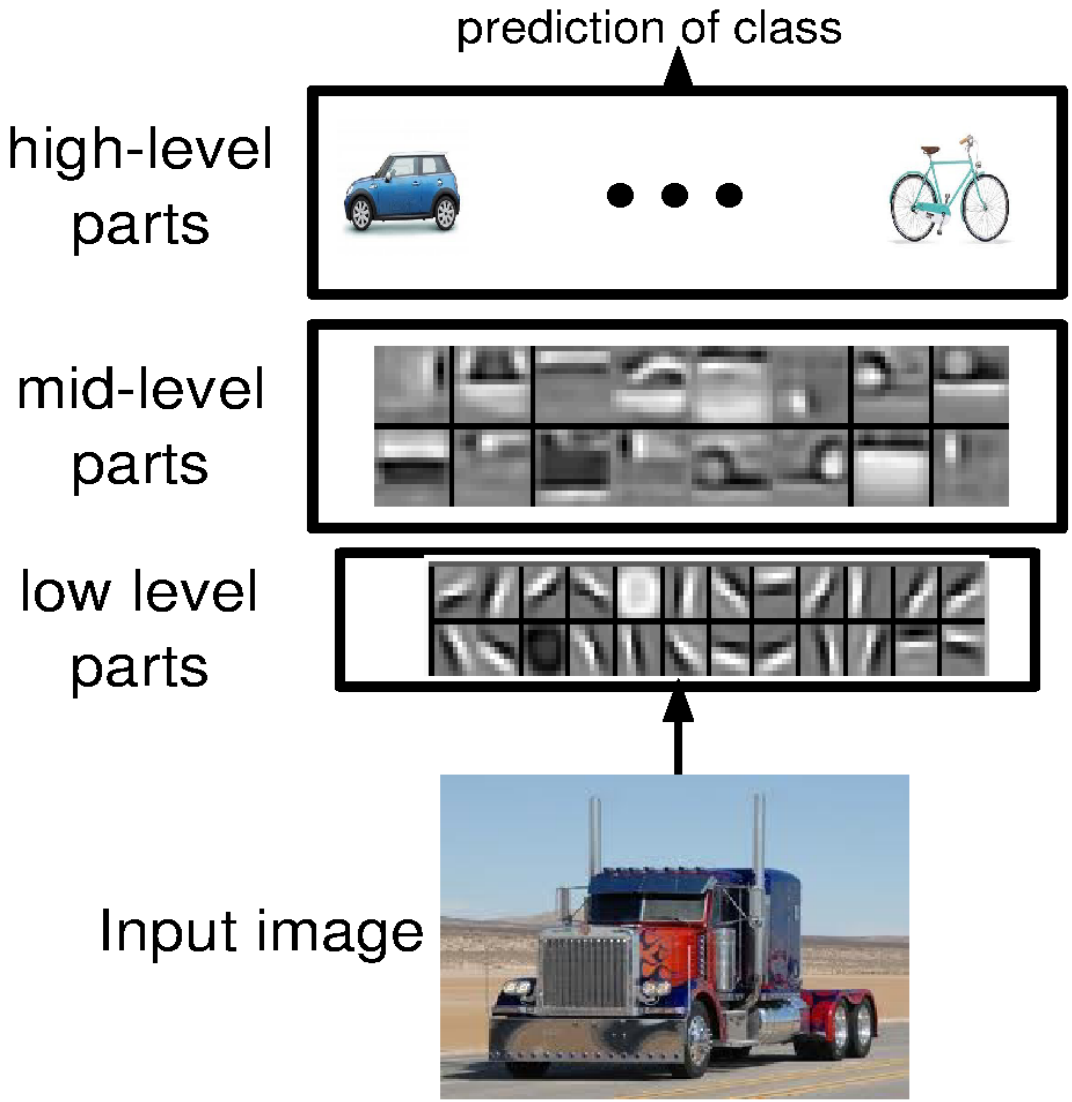
Interpretation

[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



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.