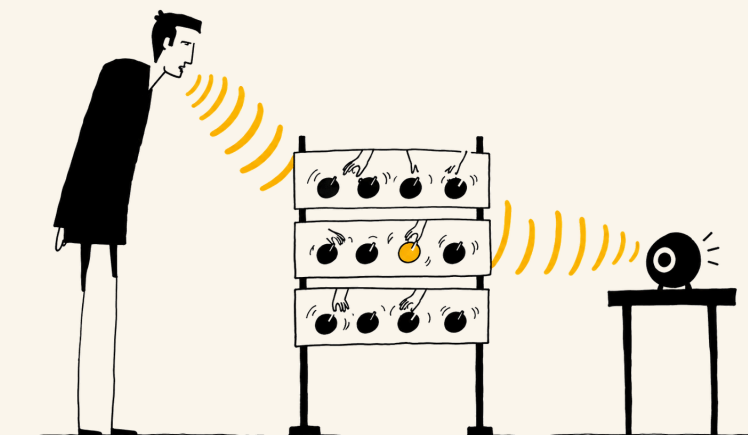


# Dive into Deep Learning



ton Zhang, Zachary C. Lipton, Mu Li, Alexander J. Smola · 20

Illustrated audio lectures

**Lecturized**

# **Dive into Deep Learning**

Illustrated audio lectures

Aston Zhang, Zachary C. Lipton, Mu Li, Alexander J. Smola · 2023

Retold as illustrated lectures by Lecturized  
[lecturized.com](https://lecturized.com)

From the original: Dive into Deep Learning (Cambridge University Press, 2023), [d2l.ai](https://d2l.ai)

# Contents

Programs that learn from examples

What every learning system is made of

Scoring the model, and the exam it must pass

How much, or which one?

Many labels, rankings and sequences

Learning without a boss, and a world that changes

Learning by acting, and where it all began

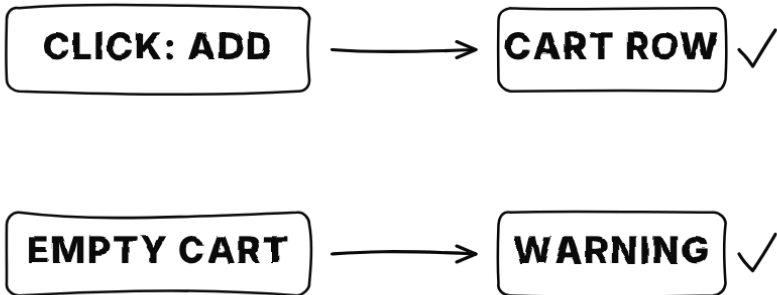
What changed: data, chips and tools

Where it already works, and what makes it deep

# Programs that learn from examples

Hold your phone a little way from your mouth and say the words that wake its assistant, "Hey Siri" or "OK Google". Then say something close but wrong, like "Hey Siria" or "OK Doodle", and watch what happens. Most of the time the phone wakes for the real words and stays asleep for the near misses, even in a noisy kitchen and even when someone else speaks. That small trick hides a question this whole book is built to answer: how did anyone write that program?

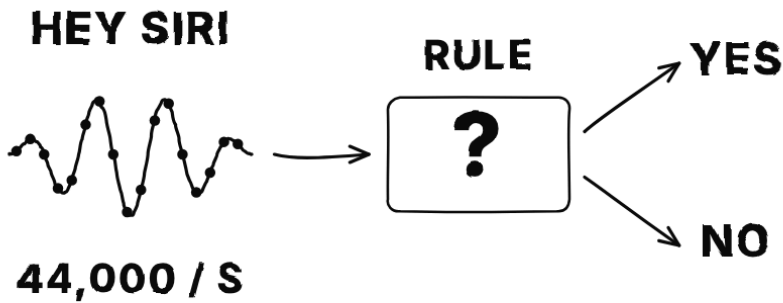
## RULES



A shop runs on rules

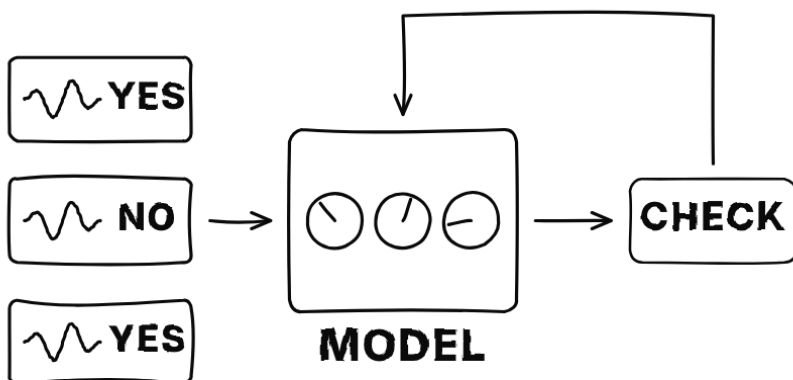
For most of the history of computing, a program was a rigid list of rules written by people. Think of an online shop: when a customer adds an item to the cart, the program writes a line linking that customer to that product, and the developers think through every odd case, such as someone pressing "buy" with an empty cart. Such a program can be planned at a whiteboard and launched with confidence before a single real customer arrives. It is a remarkable human achievement, and when rules of this kind work every time, the authors say plainly that you need no machine learning at all.

Now try to write the rules for the wake word. The microphone records roughly forty-four thousand numbers every second, each one the height of the sound wave at that instant. Which rule turns a second or two of those numbers into a confident yes or no? Nobody knows; the authors admit they could not write it either. The same wall stands in front of many tasks: forecasting tomorrow's weather, answering a question typed in free words, finding every person in a photograph, or suggesting a product you would enjoy but would never find yourself. Sometimes the right answer keeps changing with the world, and sometimes the path from pixels or sound to a meaning is far too tangled for anyone to spell out, even though our own brains walk it without effort. The spam filter in your e-mail faces both problems at once, because spammers change their tricks every week.



No rule for a voice

Here is the way out, and it is the central idea of the chapter. You cannot write the rule for recognising "Hey Siri", but you can recognise it yourself. So people collect a huge set of recordings and mark each one: this one contains the wake word, this one does not. Then, instead of a fixed program, they build a flexible one whose behaviour depends on a great many adjustable numbers called parameters, which you can picture as knobs. With the knobs set at random, the program says yes and no by chance. A learning algorithm then turns the knobs, little by little, so that the program's answers on the marked recordings get better, and it keeps going until the program is good. Once the knobs are fixed, the program is called a model.



Turning the knobs

The authors call this programming with data. You do not tell the machine how to hear the wake word; you show it many examples and let the knobs find the setting that works. The same kind of model, with another setting of the knobs, could be taught a different word just as well, but a very different job, such as writing captions for photographs or translating English into Chinese, may need a different kind of model altogether. The voice assistant on your phone, the predicted travel time in your maps app and the feed that guesses which video you will watch next were all made this way, from examples rather than from hand-written rules.

This is what the book means by machine learning: programs whose performance improves with experience, usually in the form of data.

The online shop's rules never get better on their own, however many customers pass through; a learning program does. Deep learning, the subject of the book, is one family of methods inside machine learning, and today it drives much of the progress in recognising images and speech, in understanding language, in medicine and in reading genomes. The chatbot you type questions into is a deep learning model whose knobs were set by showing it an enormous amount of text. This part has left out the details of how the knobs are turned and how the quality of a model is measured; those come next.

### **Try it yourself**

Ask a friend to say your phone's wake word quietly from across the room, then a near miss in the same voice. Count how often the phone wakes for each. You should see that it usually wakes for the real word and rarely for the near miss, though not perfectly: a learned model is right most of the time, not always, because it learned from examples rather than from a rule.

## Remember

Some tasks can be solved with rules written by people, and then machine learning is not needed. Other tasks, like hearing a wake word or recognising a face, are easy for us but impossible to put into rules. For those, we collect many marked examples and let a learning algorithm adjust the knobs of a flexible program until it does the job well. The tuned program is called a model, and this way of working is programming with data. Deep learning is one powerful family of these methods, and it is what the rest of the book teaches.

---

## Questions

1. Why is a wake word such as "Hey Siri" hard to recognise with hand-written rules?
  - a. The microphone collects too few numbers to tell words apart
  - b. No one can say which rule turns tens of thousands of sound measurements into yes or no, even though people hear the word easily
  - c. Phones are not fast enough to follow a long list of rules
  - d. The wake word changes every time the phone is updated
2. When does the book say you do not need machine learning at all?
  - a. When the task involves a large database of customers
  - b. When the program has to run on a phone

- c. When you can write rules that work every time, as for an online shop's cart
  - d. When the data are recorded as numbers rather than words
3. A learning algorithm has just started on a fresh program whose knobs are set at random. What does the program do?
- a. It answers yes and no roughly by chance
  - b. It already recognises the wake word but slowly
  - c. It refuses to answer until it has seen all the data
  - d. It recognises every English word equally well
4. What makes a learning program different from the online shop's program as time goes on?
- a. The shop's program is faster because its rules are fixed
  - b. The learning program gets better with more experience, while the shop's rules stay the same until developers rewrite them
  - c. The learning program needs no data once it is launched
  - d. The shop's program learns from each customer's clicks

## Answers

1. **b)** The phone records about forty-four thousand numbers a second, and nobody knows a rule that maps them reliably to yes or no, although our own ears do it without effort.
2. **c)** If rules written by people work in every case, as they can for the shop's cart, there is nothing for learning to add.
3. **a)** With random knobs the program is useless; only turning the knobs to do better on the marked examples makes it good.

**4. b)** Machine learning means performance that improves with experience; the shop's rules never improve on their own, however many customers pass through.

## Practice

**1.** The book lists tasks that even the best programmers would struggle to write rules for, such as forecasting tomorrow's weather or suggesting products you would enjoy. Pick two of them and explain, for each, which of the two reasons applies: the right answer keeps changing with the world, or the path from input to meaning is too tangled to spell out.

**Hint:** Ask yourself for each task: would a perfect rule written today still be right next year? And could you describe step by step how you do it in your head?

**Answer:** Product suggestions fall under the first reason: people's tastes and the products on offer change, so any fixed rule goes stale and the solution must adapt. Finding every person in a photograph falls under the second: we do it effortlessly, but the steps from pixels to 'this is a person' are beyond what anyone can write down. Weather forecasting has a bit of both.

**2.** The book describes how one could "program" a cat detector without writing any rule about whiskers or ears. Explain in your own words what you would give the system and what it would learn to output.

**Hint:** Think of the wake-word recipe: marked examples in, knobs adjusted, a model out.

**Answer:** You give it many pictures marked as cats or dogs. A learning algorithm adjusts its knobs until it outputs a large positive number for a cat, a large negative number for a dog, and something near zero when it is

unsure. The behaviour comes from the examples, which is what the book calls programming with data.

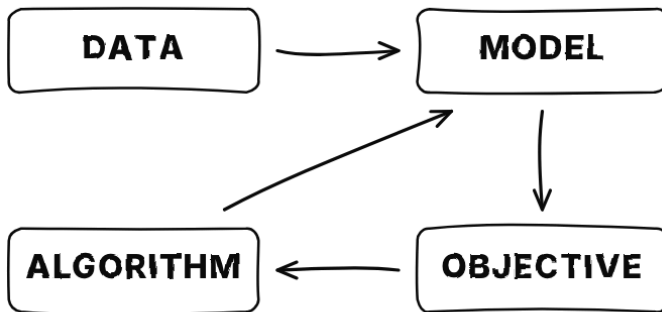
**3.** A model has been trained to wake for "Alexa". The makers now want it to wake for "Apricot" instead, and separately want a program that writes captions for photographs. For which of the two can they probably keep the same kind of model, and why?

**Hint:** Compare what goes in and what comes out in each job.

**Answer:** For "Apricot": it is the same kind of job, audio in and yes or no out, so the same family of models with another setting of the knobs should do. Captioning photographs takes images in and gives sentences out, a fundamentally different job that may need a different kind of model altogether.

# What every learning system is made of

Take any photograph on your phone and zoom in until it breaks into little squares. Each square, a pixel, is nothing more than three numbers: how bright the red, the green and the blue light are at that spot. A small photograph two hundred pixels wide and two hundred tall is therefore a list of one hundred and twenty thousand numbers. That is what a learning program actually sees, and it is the first of four parts that every machine learning system has, whatever its job.



A photo is numbers

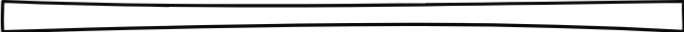
The four parts are these: the data it learns from, a model that transforms the data, an objective function that scores how well the model is doing, and an algorithm that adjusts the model's knobs to improve that score. The wake-word example from the first part had all four, and it belonged to the most common kind of problem, supervised learning, where every example comes with the answer the program should learn to give.

The data is a collection of examples, each turned into numbers. The numbers describing an example are its features, and in supervised learning the answer to be predicted is its label. For a hospital, the features of a patient might be age, pulse, medicines and recent operations, and the label whether that patient survived the next thirty days. When every example has the same number of features, the data is tidy; but photographs from the internet come in all shapes, and text is worse, since one review says only "it stinks!" while another runs for pages. Handling data of varying length gracefully is one of deep learning's great advantages over older methods. The messages a spam filter reads are exactly this kind of data: some are one line, some are long letters.

| AGE | PULSE | MEDICINES | SURVIVED? |
|-----|-------|-----------|-----------|
| 64  | 72    | 3         | YES       |
| 51  | 88    | 1         | NO        |

**FEATURES** **LABEL**

 **IT STINKS!**

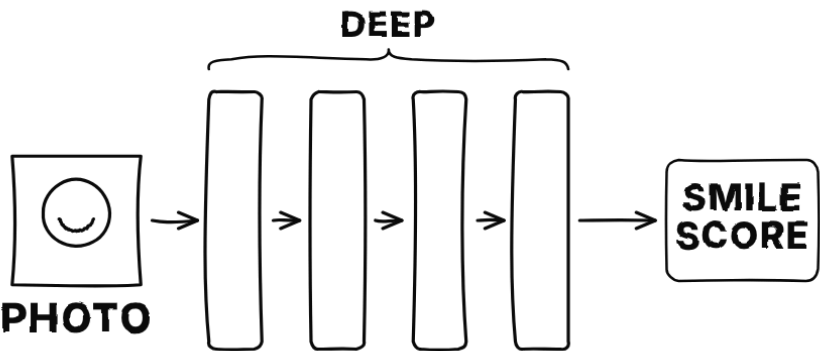


**LONG REVIEW**  
**GARBAGE IN, GARBAGE OUT**

Features and a label

Two more things about data matter enormously. First, more is better: with more examples, more powerful models can be trained, and the move from small to big data is a major reason deep learning succeeded. Second, the data must be the right data, because garbage in gives garbage out. If a group of people is missing from the examples, as with a skin cancer detector that never saw dark skin, the system fails them; and if past decisions were unfair, as with old hiring choices used to screen new applications, the model can quietly learn and repeat that unfairness, without anyone meaning it to. A recommendation feed trained only on what one group of users watched will serve everyone else badly.

The model is the machinery that takes in data of one kind and gives out predictions, perhaps of another kind: photographs in, a smile score out; sensor readings in, a warning of something abnormal out. Simple models suit simple problems, but the problems in this book are beyond them. Deep learning stands apart because its models are chains of many transformations, one after another, each passing its result to the next, and that depth is what gives deep learning its name. How the objective function and the learning algorithm work is left for the next part.



A chain of transformations

## Remember

Every machine learning system has four parts: data, a model, an objective function and an algorithm that improves the model. Data are examples turned into numbers, described by features, and in supervised learning each carries a label. More data helps, but wrong or one-sided data leads to failure and can repeat old unfairness. A deep model is a long chain of transformations of the data, and that chain is what makes it deep.

---

## Questions

1. A team builds a hospital model that predicts whether a patient survives the next thirty days. Which of these is the label?
  - a. The patient's age and pulse
  - b. Whether the patient survived the thirty days
  - c. The list of the patient's medicines
  - d. The number of patients in the records
2. Why does text give learning systems more trouble than pictures from a single standard microscope?
  - a. Text contains no numbers at all
  - b. Reviews and messages vary hugely in length, so they do not fit a fixed number of features
  - c. Text is always full of mistakes
  - d. Microscope pictures need no features

3. A resume-screening model is trained on a company's past hiring decisions, which favoured one group. What is the danger?
  - a. The model will be too slow to use
  - b. The model will refuse to rank anyone
  - c. The model can learn and repeat the old unfairness, even if nobody intends it
  - d. The model will need fewer examples than usual
4. What makes a deep learning model 'deep'?
  - a. It is a long chain of transformations, each passing its result to the next
  - b. It is trained on very large amounts of data
  - c. It has the most accurate objective function
  - d. It works only on photographs

## Answers

1. **b)** The label is the answer the model learns to predict; age, pulse and medicines are features that go in.
2. **b)** Tidy data has the same number of features for every example; one review says 'it stinks!' and another runs for pages, and deep learning's strength is handling such varying length.
3. **c)** Garbage in, garbage out: if the examples reflect past prejudice, the model quietly learns it.
4. **a)** The depth is the number of successive transformations chained together, and that is where the name comes from.

## Worked examples

**A photograph as numbers** A colour photograph is 200 pixels wide and 200 pixels tall. Each pixel is stored as the brightness of red, green and blue light. How many numbers does a learning program receive for this one photograph?

Count the pixels: 200 across times 200 down gives 40,000 pixels.

Each pixel carries three numbers, one for each colour of light.

So the photograph is 40,000 times 3 numbers.

**Answer:** 120,000 numbers

## Practice

**1.** A colour photograph is 300 pixels wide and 200 pixels tall, stored in the same way as red, green and blue brightness for every pixel. How many features does it give the model?

**Hint:** First count the pixels, then remember how many numbers each pixel carries.

**Answer:** 180,000. There are 300 times 200, that is 60,000 pixels, and each gives three numbers.

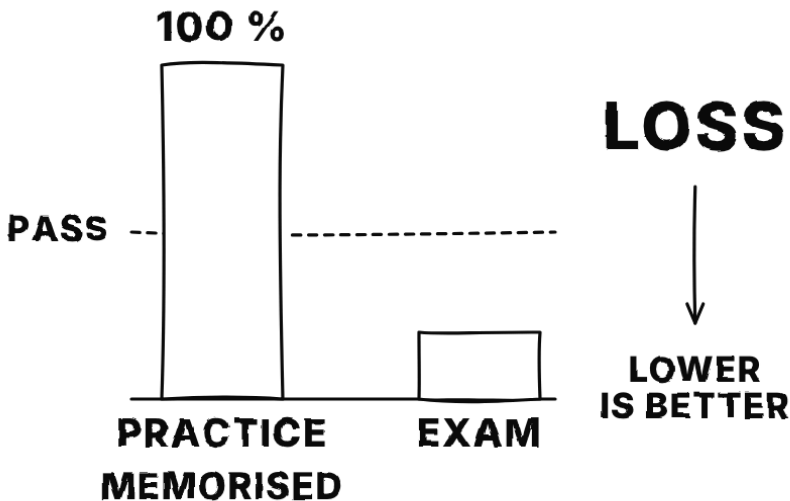
**2.** A skin cancer detector works well in tests, yet fails on many patients once it is used in a clinic. Using the idea of 'the right data', give a likely reason and say what would have to change.

**Hint:** Ask who was, and who was not, among the examples the detector learned from.

**Answer:** The training examples probably left out some group of patients, for instance people with dark skin, so the detector never learned what their skin looks like. More data of the same kind would not help; the examples must include the people the system will be used on.

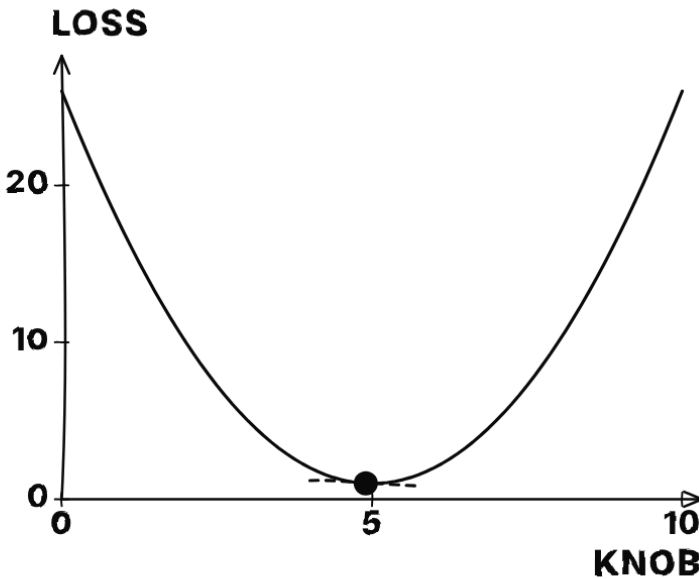
# Scoring the model, and the exam it must pass

Picture a student preparing for a final exam with a stack of practice papers. After a week, they score full marks on every practice paper. Then the real exam arrives with questions they have never seen, and they fail. They had not learned the subject; they had memorised the practice answers. Learning machines fall into exactly the same trap, and the book's way of guarding against it is the heart of this part.



Practice is not the exam

First, a machine cannot improve unless someone says what improving means. So every learning system has an objective function, a single number that scores how badly the model is doing, and by convention lower is better, which is why it is also called a loss. When the model predicts a number, the usual loss punishes the size of each miss, with big misses counting far more than small ones. When it predicts a category, the natural score is the share of examples it gets wrong, though that score is awkward to improve directly, so in practice a smoother stand-in is used. For a spam filter, the loss could simply be how many messages it put in the wrong folder.

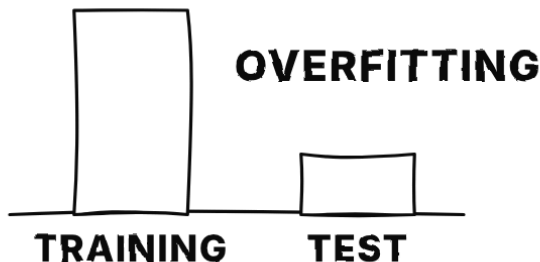
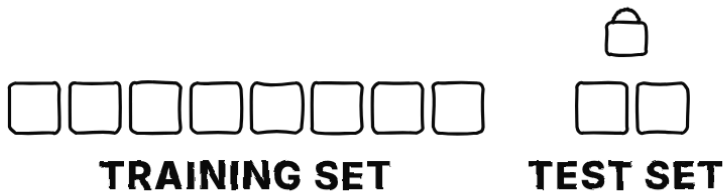


Walking downhill

Next comes the learning algorithm, the part that turns the knobs. The method behind almost all of deep learning is called gradient descent.

At each step it asks, for every knob, whether nudging it a tiny bit would make the loss go up or down, and then moves every knob a little in the direction that lowers it. Step after step, like walking downhill in fog by feeling the slope under your feet, the model gets better on its examples.

And now the trap. Doing well on the examples used for training does not prove the model will do well on new ones. So the data is split in two: a training set, on which the knobs are tuned, and a test set, kept aside and shown to the model only at the end, like the real exam. A model that shines on the training set and fails on the test set is overfitting: it has memorised instead of learning.



Training set and test set

This is supervised learning, the kind behind most of machine learning's successes in industry. Collect examples, get the correct answer for each, either from records already kept or from people who mark them by hand, feed them to the learning algorithm, and out comes a model. Then give the model inputs it has never seen and use its outputs as predictions: cancer or not from a scan, a French sentence from an English one, a share price from a company's report. When you mark a message as spam, you are adding one more labelled example to someone's training set. How the different kinds of answers, numbers or categories, change the problem is left for the next part.

### **Remember**

A model can only improve against a score, the loss, which says how badly it is doing; lower is better. Gradient descent improves the model by nudging every knob slightly in the direction that lowers the loss. Good scores on the training examples prove little, so a separate test set is kept back, like a real exam. A model that does well in training and badly on new data is overfitting. Supervised learning means learning from examples with known answers and then predicting answers for new inputs.

---

## **Questions**

1. A model scores almost perfectly on its training set but badly on the test set. What has happened?

- a. The test set was too small to matter
  - b. The model has overfitted: it memorised its examples instead of learning
  - c. The loss was defined so that higher is better
  - d. Gradient descent moved the knobs in the wrong direction
2. Why is the test set kept away from the model until the end?
- a. So that it shows how the model does on data it has never seen
  - b. Because the test set has no labels
  - c. Because gradient descent cannot work with more data
  - d. So that training finishes sooner
3. What does gradient descent do at each step?
- a. It tries random settings of all the knobs and keeps the best one
  - b. It adds more examples to the training set
  - c. It checks how a small nudge to each knob would change the loss and moves the knobs the way that lowers it
  - d. It removes the examples the model gets wrong
4. A model predicts house prices. Under the usual loss for numbers, which is punished more heavily?
- a. Ten small misses of the same total size
  - b. Both equally, since only the total matters
  - c. Neither, because prices are categories
  - d. One big miss

## **Answers**

1. **b)** Shining on the training examples and failing on new ones is exactly overfitting, like the student who memorised the practice papers.
2. **a)** The test set plays the role of the real exam; if the model trained on it, it could no longer tell learning from memorising.
3. **c)** Like walking downhill in fog, it feels the slope for every knob and takes a small step downhill.
4. **d)** The usual loss for numbers makes big misses count far more than small ones.

## Worked examples

**Error rate of a spam filter** A spam filter is tested on 500 messages it never saw in training and puts 15 of them in the wrong folder. What is its error rate on the test set?

The error rate is the share of examples on which the prediction disagrees with the true answer.

Here 15 of 500 messages were wrong.

15 divided by 500 is 0.03.

**Answer:** 0.03, that is 3 per cent

## Practice

1. Under squared error, a prediction that misses by 2 costs 2 times 2, that is 4. How much does a miss of 6 cost, and how many times more is that than a miss of 2?

**Hint:** Square the difference between the prediction and the true value.

**Answer:** 36, which is 9 times as much as 4: tripling the miss multiplies the cost by nine, which is why big misses count far more.

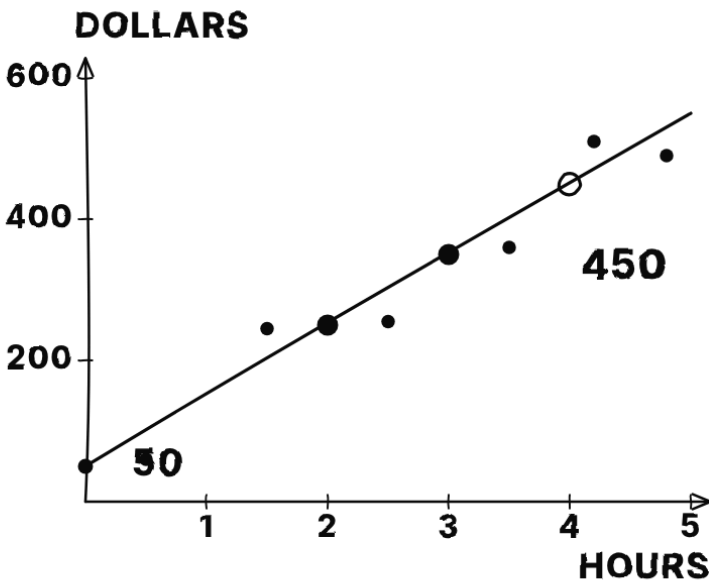
**2.** Two models are trained to recognise wake words. Model A gets 99 per cent right on the training set and 70 per cent on the test set; model B gets 92 per cent on training and 91 per cent on test. Which would you put in a phone, and why?

**Hint:** Which number tells you how the model will do on voices it has never heard?

**Answer:** Model B. The test set stands for new data, and B does far better there; A's high training score with a much lower test score is a sign of overfitting.

## How much, or which one?

Imagine a plumber cleans your drains for three hours and sends a bill for three hundred and fifty dollars. Your friend calls the same plumber, who works for two hours, and the bill is two hundred and fifty. A neighbour asks what a four-hour job would cost. Most people work it out without a pen: one more hour added a hundred dollars, so the plumber charges a hundred an hour plus a fifty-dollar call-out fee, and four hours should come to four hundred and fifty. If you followed that, the book says, you already understand the idea behind the simplest kind of supervised learning.

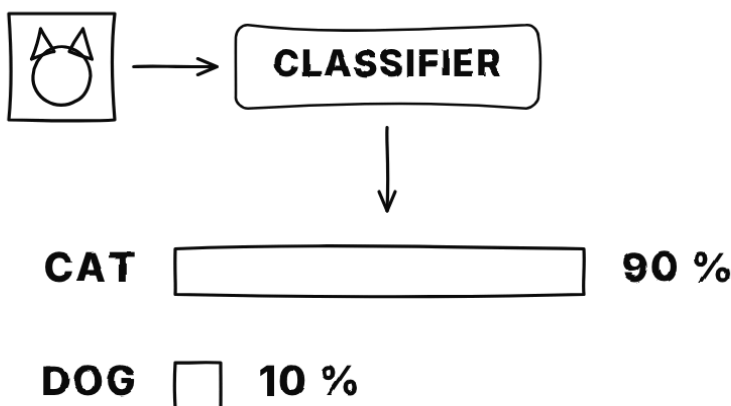


The plumber's price line

That kind is called regression, and what makes a problem regression is the answer it asks for: a number that can take any value. The examples are usually rows of a table, and each row's features form a feature vector. For houses, the row might hold the floor area, the bedrooms, the bathrooms and the minutes on foot to the town centre, and the label is the price the house sold for. A good rule of thumb is that any "how much?" or "how many?" question is regression: how many hours an operation will take, how much rain will fall in the next six hours, how many days a patient will stay in hospital. The star rating a streaming service guesses you would give a film is a regression too; the book recalls that predicting such ratings well in 2009 could win the million-dollar Netflix prize.

With the plumber, two examples fixed the price exactly. Real data is never that clean: other things the features do not capture push the answers about. So the model is not asked to hit every example, only to come as close as possible on the whole, and the usual measure of closeness is the squared miss from the last part.

The other great family is classification, and its question is "which one?". A bank's app photographs a cheque and must decide, for each handwritten mark, which character it is; a digit reader chooses among ten classes, nought to nine; the simplest case has only two classes, such as cat or dog. Firm yes-or-no answers turn out to be hard to train, so a classifier usually gives probabilities instead: shown a photo, it might say it is ninety per cent sure the animal is a cat. That number also tells you how unsure the model is, which matters a great deal for what happens next. Your spam filter is a classifier with two classes, spam and not spam, and it too works with how sure it is before it moves a message.

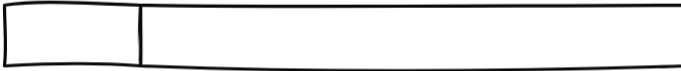


A classifier gives chances

The book makes the point with a mushroom found in a garden. A classifier trained on photographs says there is a twenty per cent chance it is a death cap, the deadliest mushroom there is. So it is eighty per cent sure the mushroom is safe, and you would still be a fool to eat it. The most likely answer is not always the one to act on: a decision has to weigh the chances against what each outcome costs, and a small chance of death outweighs any dinner. The mushroom was in fact a death cap.

**DEATH CAP 20 %**

**SAFE 80 %**



**EAT**

**THROW AWAY**

**RISK: DEATH**

**COST: DINNER**



Weigh chance and cost

Not all mistakes are equal in another way too. When classes form a family tree, like Linnaeus's tree of living things, mistaking a poodle for a schnauzer is a small slip, while mistaking a poodle for a dinosaur is absurd, so a model can be built to prefer near misses. But which tree matters depends on the use: a rattlesnake and a harmless garter snake are close relatives, yet confusing them can be fatal. The many-label and more complex kinds of prediction are left for the next part.

## Try it yourself

Write down the plumber's two bills, three hours for three hundred and fifty dollars and two hours for two hundred and fifty. Find the price per hour from the difference, then the fixed fee, and predict a five-hour job. You should get one hundred per hour, fifty to show up, and five hundred and fifty dollars: you have just fitted a model to data and used it to predict.

## Remember

Regression answers "how much?" or "how many?" with a number, such as a price or a number of hours. Classification answers "which one?" by choosing among categories, such as cat or dog, or the digits nought to nine. Classifiers usually give probabilities, which show how sure they are. A decision must weigh those chances against the cost of each outcome, so an eighty per cent safe mushroom still goes in the bin. Some mistakes are worse than others, and which ones depends on how the model will be used.

---

## Questions

1. Which of these questions is a regression problem?
  - a. Is this e-mail spam or not?
  - b. Which digit is written on this cheque?

- c. How many hours will this operation take?
  - d. Is this mushroom a death cap?
2. A classifier is 80 per cent sure a mushroom is safe. Why should you still not eat it?
- a. Because 80 per cent is the least likely answer
  - b. Because the decision must weigh the 20 per cent chance against the cost, and that cost is death
  - c. Because classifiers never give probabilities
  - d. Because the classifier was trained on cats and dogs
3. Why do classifiers usually output probabilities rather than a firm answer?
- a. Firm answers are hard to train, and a probability also shows how unsure the model is
  - b. Probabilities need less data
  - c. Firm answers cannot be shown on a screen
  - d. Probabilities make every mistake equally costly
4. In a model that tells snakes apart for hikers, which mistake matters most?
- a. Calling a garter snake a grass snake
  - b. Calling one harmless snake by another harmless snake's name
  - c. Calling a rattlesnake a garter snake
  - d. Calling a garter snake a rattlesnake

## **Answers**

1. **c)** A 'how much?' or 'how many?' question asks for a number, and that makes it regression; the others choose among categories.
2. **b)** The most likely answer is not always the one to act on; a small chance of a fatal outcome outweighs the benefit of a dinner.
3. **a)** Firm yes-or-no outputs are hard to optimise, and a probability such as ninety per cent cat carries the model's uncertainty, which decisions need.
4. **c)** The two snakes are close relatives, but taking a dangerous rattler for a harmless garter snake can be fatal; which mistakes matter depends on the use.

## Worked examples

**The plumber's prices** A contractor works 3 hours and bills 350 dollars; for another customer they work 2 hours and bill 250 dollars. Assume a fixed call-out fee plus a price per hour. Find both, and predict the bill for 4 hours.

One extra hour, from 2 to 3, added 350 minus 250, that is 100 dollars, so the price per hour is 100 dollars.

Two hours cost 2 times 100, that is 200 dollars of work, so the rest of the 250-dollar bill, 50 dollars, is the fixed fee.

Check on the other bill: 50 plus 3 times 100 is 350, as billed.

For 4 hours: 50 plus 4 times 100.

**Answer:** 100 dollars per hour plus a 50-dollar fee; 450 dollars for 4 hours

**To eat or to throw away** A classifier gives a 0.2 chance that a mushroom is a death cap and 0.8 that it is safe. Throwing away a safe mushroom costs 1 (a lost dinner); throwing away a death cap costs nothing; eating a death cap costs a life, which no dinner can outweigh.

What is the expected cost of throwing it away, and which action should you take?

Throwing it away:  $0.2 \times 0$  for the death cap plus  $0.8 \times 1$  for the lost dinner.

That gives 0.8.

Eating it: a 0.2 chance of a cost without limit, so its expected cost has no limit either.

A finite cost beats an unlimited one, so throw it away.

**Answer:** Throwing it away costs 0.8 on average; eating it costs without limit, so you throw it away.

## Practice

1. A window cleaner charges 380 dollars for a 4-hour job and 140 dollars for a 1-hour job, with a fixed fee plus a price per hour. Find the price per hour, the fee, and the bill for 3 hours.

**Hint:** Three extra hours made the bill go up by how much?

**Answer:** The extra 3 hours added 240 dollars, so 80 dollars per hour; then 140 minus 80 leaves a 60-dollar fee; 3 hours cost 60 plus 240, that is 300 dollars.

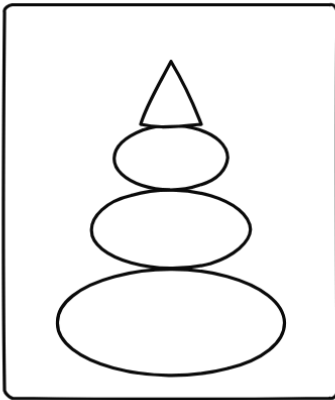
2. Sort these into regression and classification, and say why: the price a flat will sell for; whether a scan shows cancer; how much rain will fall in the next six hours; which letter a handwritten mark is.

**Hint:** Ask whether the answer is a number or a choice among categories.

**Answer:** The flat's price and the amount of rain are numbers, so they are regression; cancer or not and which letter are choices among categories, so they are classification.

## Many labels, rankings and sequences

Open the photo app on your phone and type "dog" into its search box. Up come your dog pictures, but also the one from a park where a dog, a child, a bicycle and a tree all share the frame. That photograph cannot belong to just one class. The phone has tagged it with several labels at once, and that small fact pushes us beyond the "which one?" question of the last part.



**ONE CLASS** ✕

**DONKEY** ✓

**DOG** ✓

**CAT** ✓

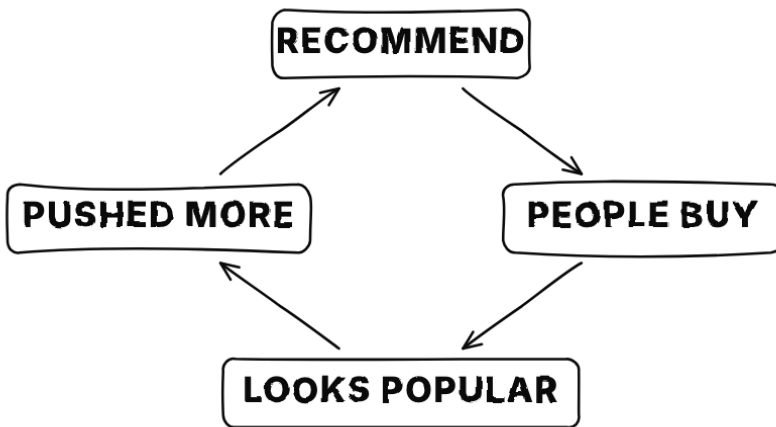
**ROOSTER** ✓

One photo, many labels

The book's picture is the fairy tale of the Town Musicians of Bremen: a donkey, a dog, a cat and a rooster standing on one another's backs.

A classifier forced to pick a single animal is bound to be wrong, because the honest answer is all four. Predicting classes that are not exclusive is called multi-label classification, and it is how automatic tagging works. A post on a technology blog might carry five to ten tags, and the tags travel together: posts about cloud computing tend to mention one big cloud provider, posts about machine learning tend to mention graphics chips. At the extreme, the United States National Library of Medicine tags every indexed medical article with terms chosen from about twenty-eight thousand headings, work that human experts take up to a year to finish, so a learned model can supply temporary tags in the meantime.

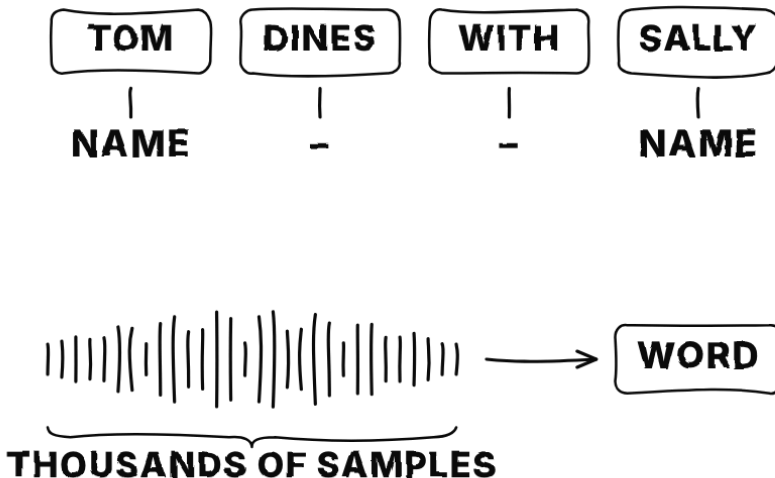
Search asks a different question again: not whether a page is relevant, but in what order the relevant pages should be shown. Google's original secret ingredient, PageRank, gave every page a score of authority that, oddly, did not depend on the query at all; a simple filter first found the relevant pages, and the score then put them in order. Today search engines learn scores that depend on the query and on how people behave.



Ranking and its loop

Recommender systems are close cousins of search, with one big difference: the list is made for you personally. A science fiction fan and a lover of old Peter Sellers comedies should see very different pages. Some feedback is explicit, like star ratings and reviews; some is implicit, like skipping a song, which may mean dislike or just the wrong moment. The feed on a video app that seems to know what you will watch next is a recommender of exactly this kind, guessing a score for each video and showing you the top ones. These systems are worth a fortune but have deep flaws. People mostly rate what they love or hate, so three-star ratings are oddly rare, and the system's own suggestions shape what people buy, so a product it pushes looks popular and gets pushed even more, a feedback loop that is still an open research problem.

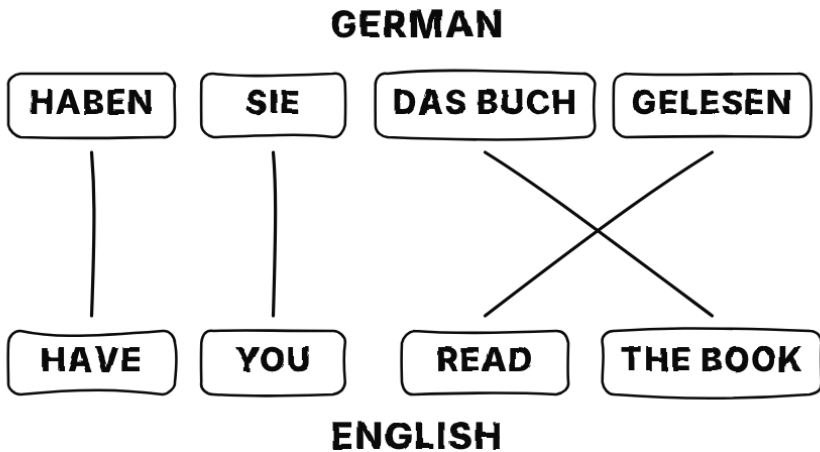
So far every example was handled alone and forgotten once answered. But a video is a run of frames, and what happens in one frame is far clearer when you see the frames around it. A patient in intensive care is watched hour after hour, and a model warning of danger must remember the whole history, not just the latest reading. Problems where the input or the output, or both, is a sequence of varying length are called sequence learning. A chatbot reads your whole message as a sequence of words and answers with another sequence, so it is sequence learning from start to finish.



How sequences line up

The book names a few kinds. In tagging, every word gets a mark, such as whether it is part of a name, so input and output line up one to one. In speech recognition, thousands of sound measurements make a

single word, so a long input becomes a short output; text to speech runs the other way, short text into long audio. When you dictate a message to your voice assistant, it does the first of these, and when it reads a reply aloud, it does the second. Machine translation is harder still, because the order changes: German likes to put the verb at the end, so a word-for-word English version would say "Have you yourself already this excellent textbook looked at?" Reading the layout of a web page, and holding a conversation that remembers what was said long ago and what the world is like, are harder yet and still being researched. Learning without labels at all is left for the next part.



Translation changes the order

## Remember

Multi-label classification lets one example carry several labels at once, which is how automatic tagging works. Search and recommendation are about putting items in order, and recommendation adds personal taste. Recommenders face skewed feedback and loops in which their own suggestions make items look more popular. Sequence learning handles inputs or outputs of varying length, where order and context matter. Tagging, speech recognition, text to speech and translation are sequence tasks that differ in how input and output line up.

---

## Questions

1. Why is an ordinary multiclass classifier a poor fit for a picture of the Town Musicians of Bremen?
  - a. The picture has too many pixels for a classifier
  - b. Animals cannot be told apart by a model
  - c. It must pick one class, but the honest answer is four animals at once
  - d. Fairy-tale pictures are never in the training data
2. What is the main difference between a recommender system and a search engine?
  - a. A recommender orders items for a particular person's taste
  - b. A recommender never gives items a score
  - c. A search engine works only with films

- d. A search engine cannot rank pages
3. A shop's recommender pushes one blender, so more people buy it, so it is pushed even more. What problem is this?
- a. Overfitting to the test set
  - b. A feedback loop, where the system's own suggestions shape the data it learns from
  - c. Multi-label classification
  - d. Too few three-star ratings
4. Which task turns a long input into a short output?
- a. Text to speech
  - b. Tagging every word of a sentence
  - c. Translating German into English
  - d. Speech recognition

## Answers

1. **c)** When several classes are true at once, the problem is multi-label classification, which lets the model say donkey, dog, cat and rooster together.
2. **a)** Both put relevant items in order, but a recommender is personal: the science fiction fan and the comedy lover see different lists.
3. **b)** The purchases partly reflect the recommendations, so the item looks better than it is and gets recommended even more.
4. **d)** In speech recognition thousands of sound measurements make one word, so the transcript is far shorter than the recording; text to speech goes the other way.

## Practice

**1.** The book tags the sentence "Tom has dinner in Washington with Sally", marking each word that is part of a name. Which words get the mark, and why are input and output said to line up in this task?

**Hint:** Look for people and places, and count how many marks there are compared with words.

**Answer:** Tom, Washington and Sally get the mark, and the other four words do not. There is exactly one mark per word, in the same order, so the output lines up one to one with the input, unlike in translation.

**2.** A music app learns from which songs people skip. Explain why a skip is weaker evidence than a one-star rating, and what that means for the recommender.

**Hint:** Think of reasons for skipping a song other than disliking it.

**Answer:** A skip is implicit feedback: it may mean dislike, or only that the song did not suit the moment. A one-star rating says plainly that the person disliked it. A recommender must therefore treat skips as uncertain hints rather than clear verdicts.

**3.** Why does a model that warns of danger in an intensive care unit need to be a sequence model rather than one that looks at each hour's readings alone?

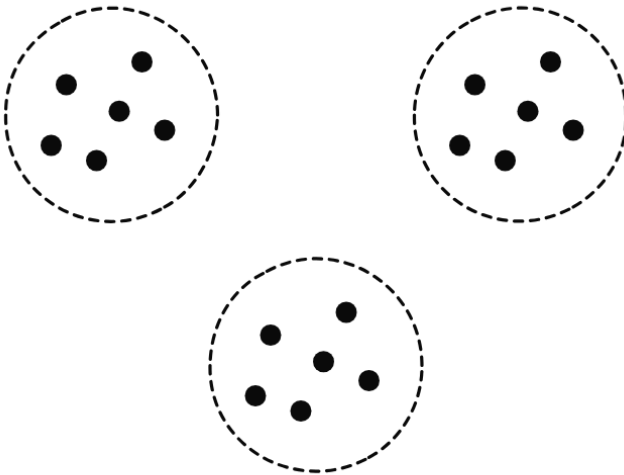
**Hint:** What does a worrying trend look like in a single reading?

**Answer:** Danger often shows in the history, such as readings drifting over many hours, which no single reading reveals. A model that forgets each hour's data cannot see that trend, so it must take the patient's whole sequence of measurements into account.

# Learning without a boss, and a world that changes

Open the photo gallery on your phone and look for its "people" or "places" albums. Nobody told the phone which faces belong to whom or what a beach looks like, yet it has sorted thousands of pictures into piles that mostly make sense. That is learning of a different kind from everything so far, because there were no labels at all.

## CLUSTERING

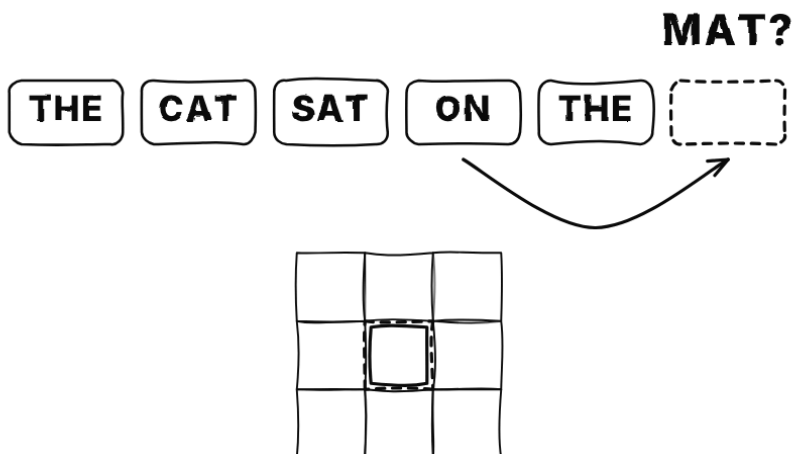


Groups with no labels

The book pictures supervised learning as a job with a bossy manager who stands at your shoulder and says exactly what to do in every case.

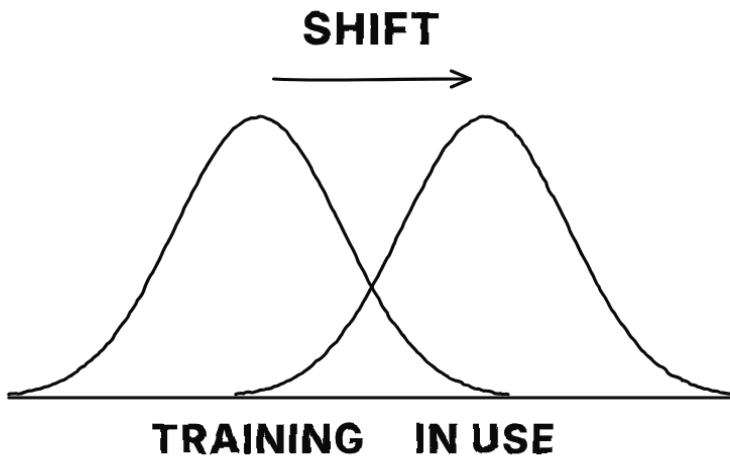
Dull, but easy: you copy the boss until you have the pattern. Now imagine a boss who dumps a mountain of data on your desk and says only, "Do some data science with it." That is unsupervised learning, and the questions you can ask are limited only by your imagination. You can sort things into groups, as the gallery does, which is called clustering. You can look for a few numbers that capture what matters, the way a tailor describes a whole body with a handful of measurements. You can place words in a space where their meanings behave like positions, so that starting from Rome, taking away Italy and adding France lands you near Paris. You can hunt for causes hidden in tables of prices, pollution and wages, or build generative models that learn what the data looks like so well that they can produce new examples of their own. The tools that make a new picture from a sentence you type are generative models of this kind.

A clever middle way is self-supervised learning, where the data supplies its own exercises. Hide a word in a sentence and ask the model to guess it from the words around it; with all the text in the world to practise on, no person has to label anything. For pictures, you can cover a patch and ask the model to fill it in. What the model learns this way is later fine-tuned for the job you actually need. The chatbot you talk to began exactly like this, by learning to guess the missing or next words in a vast amount of text.



The data sets the exercise

All these methods share a quiet assumption: the data is gathered first, the learning happens afterwards, and the model never touches the world while it learns. That is called offline learning, and it is simple, but the world is not a pile of data. Real systems act, and their actions change what they see next. Some environments remember what you did, some help you, like a person speaking clearly into a dictation app, and some fight you. Your spam filter faces an enemy: spammers study what gets blocked and change their messages to slip through. When the data the model meets in use differs from the data it trained on, we have distribution shift. The book's example is the exam written by the lecturer after a term of homework set by the assistants. How a learner can act in such a world is left for the next part.



When the data shifts

### **Remember**

Unsupervised learning finds structure in data without labels: groups, a few key numbers, hidden causes, or new examples. Self-supervised learning makes its own labels from the data, such as guessing hidden words, and then is fine-tuned for a task. Most learning is offline: the data comes first and the model never acts while it learns. When the world the model meets differs from the data it trained on, that is distribution shift.

---

## Questions

1. A phone sorts holiday photos into groups of beaches, mountains and faces without anyone labelling them. What kind of learning is this?
  - a. Supervised learning, because the groups are correct
  - b. Unsupervised learning, specifically clustering
  - c. Reinforcement learning
  - d. Regression
2. Why does self-supervised learning need no human labelling?
  - a. The data sets its own exercises, for example a hidden word that the surrounding words must reveal
  - b. It uses only small datasets
  - c. It copies labels from a supervised model
  - d. It learns only from images
3. A spam filter trained last year now lets through many new spam messages. What best explains this?
  - a. The filter was trained online
  - b. The filter uses clustering
  - c. Distribution shift: spammers changed their messages, so new data differs from the training data
  - d. The filter has too few parameters to read e-mail
4. What assumption do supervised and unsupervised learning share, according to this part?
  - a. That every example has a label

- b. That the model acts in the world while it learns
- c. That the test data is written by the same person as the training data
- d. That the data is gathered first and the model never acts while it learns

## Answers

- 1. **b)** No labels were given; finding groups of similar examples in unlabelled data is clustering, a kind of unsupervised learning.
- 2. **a)** Hiding part of the data and asking the model to guess it turns any text or image into practice material, with the hidden part as the answer.
- 3. **c)** The environment fights back; when the data met in use differs from the training data, the model's performance drops.
- 4. **d)** Both are offline learning: the pile of data comes first, and the model's outputs never change what data it sees.

## Practice

- 1. The book compares distribution shift to an exam written by the lecturer after homework set by the teaching assistants. Explain in your own words why a student who did well on the homework might still struggle, and how that maps onto a model.

**Hint:** Ask what the student practised on, and what they were tested on.

**Answer:** The student practised on questions in the assistants' style, but the exam comes in the lecturer's style, so the questions differ from what they trained on. A model is the same: it learns from training data, and if the data it meets in use comes from a different source or a changed world, its good training performance may not carry over.

**2.** For each environment, say whether it helps the learner, fights it, or neither: a person speaking slowly and clearly into a dictation app; spammers writing e-mails; the weather.

**Hint:** Ask whether the environment has a goal that involves the model.

**Answer:** The careful speaker helps, because they want to be understood. Spammers fight, adapting their messages to evade the filter. The weather does neither; it may change over time, but not in response to the model's wishes.

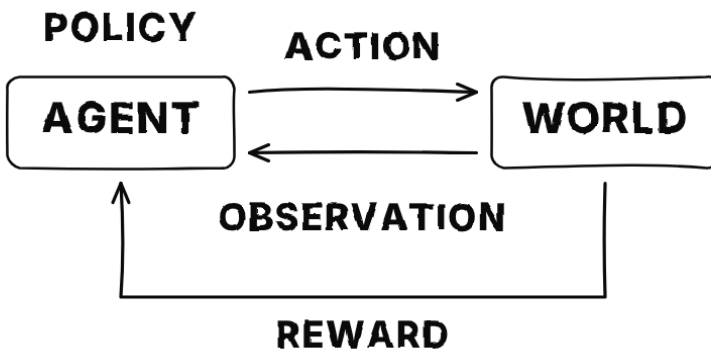
**3.** Suggest how you could turn a large collection of photographs, none of them labelled, into training exercises for a self-supervised model.

**Hint:** Hide something the photograph itself can reveal.

**Answer:** Cover a patch of each photograph and ask the model to fill it in from the rest; the original photograph supplies the correct answer, so no one has to label anything. The model learns about shapes and textures, and can later be fine-tuned for a real task.

# Learning by acting, and where it all began

Watch a child learning a video game without reading the instructions. They press buttons, the screen changes, points go up or a life is lost, and after a while they play well. Nobody told them the right button for each moment; they only saw the score. That is the idea of reinforcement learning, the kind of learning meant for things that act: robots, game players, and programs that hold a conversation.

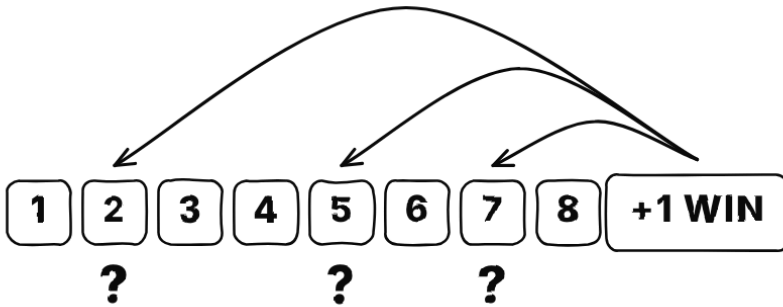


The agent's loop

The picture is a loop. An agent looks at its world, which gives it an observation; it chooses an action; the world changes and hands back a reward; and the loop goes round again. The agent's habit of turning what it sees into what it does is called its policy, and the whole aim is to learn a good one. The idea is so general that ordinary supervised learning fits inside it, with one action for each class and a reward equal to how well the guess did. The famous successes are a program that learned old Atari games from the screen alone and beat human players, and AlphaGo, which beat the world champion at Go.

What makes it hard is what the world does not say. No one tells the agent the best action, only the reward, and the reward may come long after the actions that earned it. In chess the only real reward is the win or the loss at the end, so which of forty moves deserves the credit? This is the credit assignment problem, the same one faced by someone promoted after a year of good decisions who wants to know which ones mattered. The agent may also see only part of its world, like a cleaning robot stuck in one of several identical cupboards, which must remember how it got there to know where it is. And it must keep choosing between using the best plan it knows and trying something new that might be better: exploiting or exploring. A recommendation feed lives with this choice every day: show you more of what you already like, or try something new and risk a skip. Because all of this at once is too much, researchers study simpler cases, which the book names without going into them.

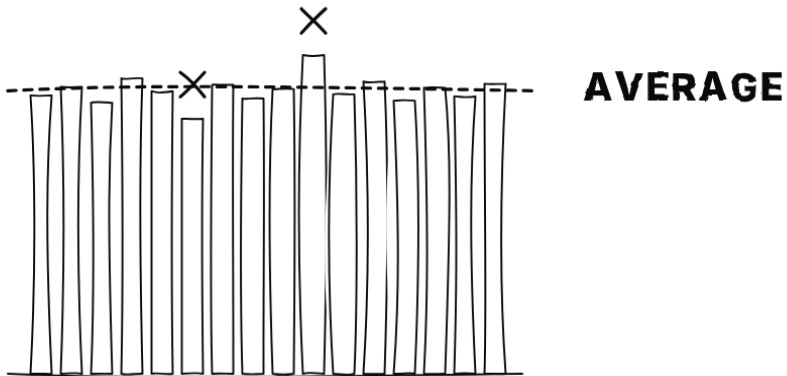
# CREDIT?



Which move earned the win?

Now the book steps back into history, because the wish to learn from data is centuries old. Gauss gave us the method of least squares, still used in insurance and medicine, and simple straight-line models describe laws such as the link between current and voltage in a wire. In the sixteenth century, the geometer Jacob Köbel described how to fix the length of a foot: stop sixteen men leaving church, line up their feet, and take the average; a later improvement sent away the shortest and the longest foot first, one of the earliest trimmed averages. Ronald Fisher built much of modern statistics, and his small dataset of iris flowers from 1936 is still used for demonstrations; he was also a supporter of eugenics, a reminder that data has long been put to bad uses as well as good. Shannon's theory of information and Turing's question, "Can machines think?", fed in too.

# 16 FEET



Köbel's foot

The brain was the other inspiration. Donald Hebb proposed that connections between neurons strengthen when they help, and that idea of rewarding good behaviour and weakening bad behaviour runs through the way neural networks are trained today. Neural networks took their name from the brain, though the resemblance has grown loose; what remains is layers of simple processing stacked on one another, and a way of adjusting every layer at once.

Yet from about 1995 to 2005 neural networks fell out of fashion. Computing power was scarce, and datasets were tiny: a collection of sixty thousand handwritten digits counted as huge. With so little data and computation, other methods, such as decision trees and kernel methods, simply worked better, trained in hours rather than weeks,

and came with solid guarantees. Many spam filters of that time were built on such classical methods rather than on neural networks. How that balance tipped is left for the next part.

### **Remember**

Reinforcement learning is for agents that act: they observe, act, receive rewards, and learn a policy. It is hard because the best action is never shown, rewards come late, the world may be only partly visible, and the agent must balance trying new things against using what works. Learning from data is centuries old, from averaging feet and Gauss's least squares to Fisher's statistics and Turing's question. Neural networks borrowed their name and some ideas from the brain. Around 1995 to 2005 they lost out to other methods, because data and computing power were too scarce.

---

## **Questions**

1. What makes reinforcement learning harder than supervised learning?
  - a. The agent is never shown the best action, only a reward that may come much later
  - b. The agent has no observations at all
  - c. Rewards are given for every possible action at once
  - d. It cannot be used for games

2. A chess program wins a game after forty moves. Which problem does it face in learning from this win?
  - a. Distribution shift
  - b. Partial observability
  - c. Credit assignment: working out which of the moves deserve the credit
  - d. Overfitting to the test set
3. A music app keeps playing only the genre you like most and never tries anything else. In reinforcement learning terms, what is it doing?
  - a. Exploring
  - b. Exploiting the best known strategy and never exploring
  - c. Assigning credit
  - d. Training offline
4. Why did other methods beat neural networks between about 1995 and 2005?
  - a. Neural networks had not yet been invented
  - b. Other methods needed more data
  - c. Neural networks could not be trained with rewards
  - d. Data and computing power were scarce, and other methods trained faster with solid guarantees

## **Answers**

1. a) In supervised learning every example carries the right answer; here the world gives only rewards, often long after the actions that earned them.

**2. c)** The only reward comes at the end, so the learner must decide which earlier moves earned it.

**3. b)** Using the best plan it knows is exploiting; it gives up the chance of finding something better by never exploring.

**4. d)** With small datasets and little computing power, decision trees and kernel methods worked better and took hours rather than weeks.

## Worked examples

**A trimmed average of feet** In the spirit of Köbel's method, six men's feet are measured: 26, 27, 28, 27, 31 and 21 centimetres. Find the plain average, then the trimmed average after sending away the shortest and the longest foot.

Plain average: the six lengths add up to 160 centimetres, and 160 divided by 6 is about 26.7 centimetres.

To trim, remove the shortest foot, 21, and the longest, 31, because unusual feet would pull the average about.

The remaining four add up to 26 plus 27 plus 28 plus 27, that is 108 centimetres.

108 divided by 4 is 27 centimetres.

**Answer:** Plain average about 26.7 cm; trimmed average 27 cm

## Practice

**1.** Five measurements of a foot are 25, 29, 28, 27 and 36 centimetres. Find the trimmed average after removing the shortest and the longest, and say why it is a better estimate here than the plain average.

**Hint:** Remove 25 and 36 first, then average what is left.

**Answer:** 28 cm: 29 plus 28 plus 27 is 84, and 84 divided by 3 is 28. The plain average, 29 cm, is pulled up by the one unusually long foot, which

the trimmed average ignores.

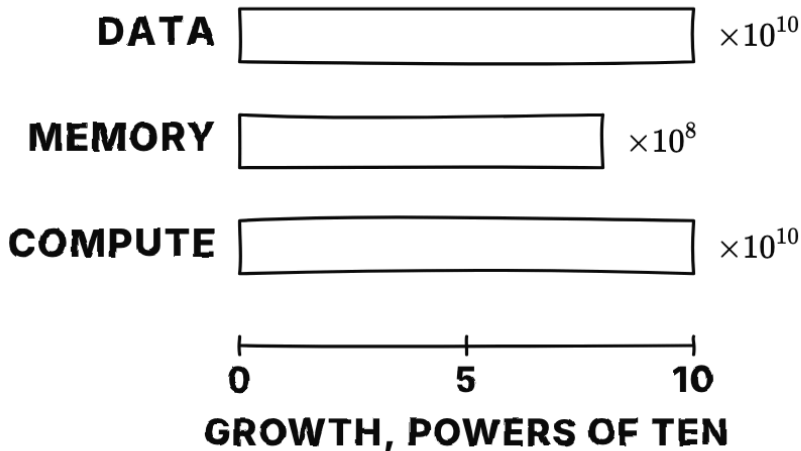
**2.** The book describes a cleaning robot trapped in one of several identical closets. Explain why looking around the closet is not enough for the robot to know where it is, and what it needs instead.

**Hint:** What would the robot see in each of the closets?

**Answer:** Every closet looks the same from inside, so the current observation does not reveal the robot's state; this is partial observability. The robot must use its earlier observations, how it moved before it entered, to work out which closet it is in.

# What changed: data, chips and tools

Think of the photos on your phone: thousands of them, each several million pixels, stored at almost no cost and backed up to the cloud without a thought. Thirty years ago a single computer's memory could not have held one of them. That change, in how much data exists and how cheaply it can be kept and processed, is what brought neural networks back from the cold.

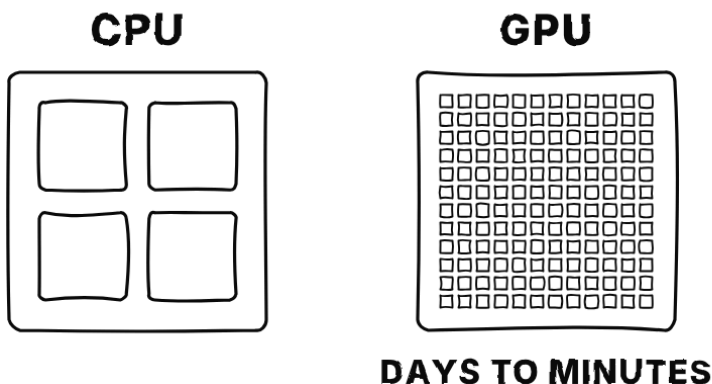


Growth from 1970 to 2020

The book shows it with a table, one row per decade. In the nineteen-seventies a typical dataset was about a hundred examples, like

Fisher's irises; by the twenty-twenties a social network's data ran to around a trillion. Memory grew too, but far more slowly: from about a kilobyte to about a hundred gigabytes. Computing power grew fastest of all, from a hundred thousand calculations a second to about a million billion. The conclusion is the interesting part: since computing has outrun memory, the best models are those that store little but compute a lot, and that is exactly what deep neural networks do. Many old ideas were therefore rediscovered in the last decade, after lying unused for years.

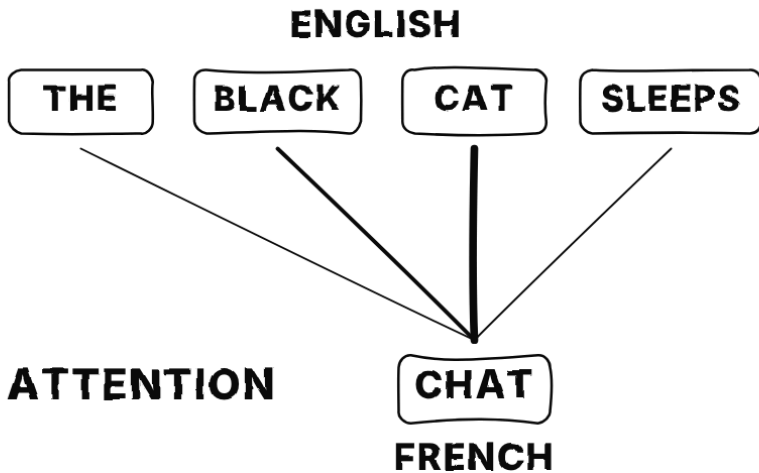
The decisive machine was the graphics card. GPUs were designed to draw video games, doing huge numbers of simple calculations side by side, and that turned out to be just what training a neural network needs. Suddenly models that had looked impossibly expensive were within reach. Splitting the work across many cards brought training times for a standard image model down from days to under seven minutes.



Many small workers

But it was not only more power poured onto old recipes; a burst of new ideas came too, which the book compares to the sudden flowering of life in the Cambrian period. Dropout adds random noise inside a network while it trains, which helps against overfitting. Attention lets a model point back to the parts of its input that matter right now, instead of cramming a whole sentence into memory before translating it. The Transformer, built from attention alone, keeps improving as data, model size and computing grow, and a single Transformer has learned to chat, caption images, play Atari and steer a robot. Language models, which predict text from text, gained surprising abilities as they were scaled up. ChatGPT, which you may use yourself, is such a language model, tuned to follow what people actually ask of it. Generative adversarial networks and, more recently,

diffusion models learned to produce realistic images; diffusion models build a picture by removing noise step by step. The apps that draw a picture from a sentence you type use diffusion models of this kind.



Attention points back

The last ingredient is software. Free deep learning frameworks, from early ones like Theano and Torch to today's PyTorch, TensorFlow and JAX, handle the hard mathematics and the hardware for you. The book gives a measure of how much that changed: in 2014, training a simple classifier was set as a real homework problem for new doctoral students at Carnegie Mellon University; today it takes fewer than ten lines of code. This part has left out the details of each method, which later chapters cover.

## Remember

Neural networks came back because data became plentiful and storage and computing became cheap. Computing power grew faster than memory, which favours models that store little and compute a lot, like deep networks. Graphics cards built for games turned out to be ideal for training them. New ideas, such as dropout, attention, Transformers, language models and diffusion models, drove progress as much as raw power. Free frameworks made it all easy enough for any programmer to use.

---

## Questions

1. Computing power has grown much faster than memory. What kind of model does that favour?
  - a. Models that store a great deal but compute little
  - b. Models that store little but compute a lot, like deep neural networks
  - c. Models that need no data
  - d. Models that run without a computer
2. Why did graphics cards turn out to be so useful for deep learning?
  - a. They have more memory than any other chip
  - b. They were designed for neural networks from the start
  - c. They do huge numbers of simple calculations side by side, which is what training needs

- d. They store datasets more cheaply
3. What does attention let a translation model do?
- a. Point back to the parts of the input that matter now, instead of holding the whole sentence in memory first
  - b. Add random noise to prevent overfitting
  - c. Generate images from noise
  - d. Train on many graphics cards at once
4. According to the book, what made deep learning accessible to ordinary programmers?
- a. Smaller datasets
  - b. The end of overfitting
  - c. Reinforcement learning
  - d. Free frameworks that handle the mathematics and the hardware

## Answers

1. **b)** When calculation is cheap and memory relatively scarce, the sweet spot moves to models that spend many calculations on few stored numbers, which is what deep networks do.
2. **c)** GPUs were built to draw games by doing many simple calculations at once, and training a neural network is exactly that kind of work.
3. **a)** With attention the model need not cram a whole sentence into fixed memory; it looks back at the relevant parts as it goes.
4. **d)** A task that was a doctoral homework problem in 2014 now takes fewer than ten lines of code, thanks to frameworks.

## Worked examples

**Many cards, one big batch** Training is spread over 1,024 graphics cards, and each card handles a small batch of 32 images at a time. How many images are processed together in one step across all the cards?

Each card takes its own batch of 32 images.

All 1,024 cards work at the same time, so the batches add up.

1,024 times 32 is 32,768.

**Answer:** 32,768 images, about 32,000

## Practice

**1.** In the book's table, a typical dataset grew from 100 examples in 1970 to 1 trillion in 2020, while memory grew from 1 kilobyte to 100 gigabytes. Taking a gigabyte as a million kilobytes, by what factor did each grow, and which grew more?

**Hint:** A trillion is a 1 followed by twelve zeros; 100 gigabytes is 100 million kilobytes.

**Answer:** Data grew by a factor of ten billion (from 100 to a trillion); memory by a factor of a hundred million (from 1 to 100 million kilobytes). Data grew about a hundred times more, which is why memory has not kept pace and models must be memory-efficient.

**2.** If 512 cards each took a batch of 64 images, how many images would one step process together, and how does it compare with 1,024 cards taking 32 each?

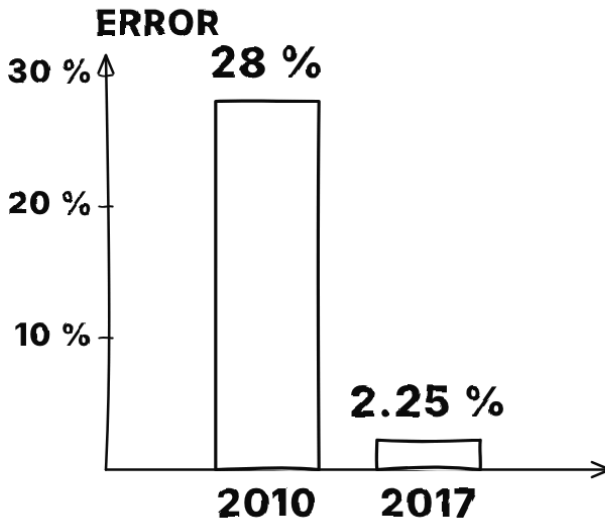
**Hint:** Multiply the number of cards by the batch on each card.

**Answer:** 512 times 64 is 32,768 images, exactly the same as 1,024 times 32: halving the cards while doubling each card's batch leaves the total unchanged.

## **Where it already works, and what makes it deep**

Pay for something online today and, in the second before the payment goes through, a model checks it for fraud. Post a letter and, in many countries, a machine reads the handwritten address; that very task gave us the famous collection of handwritten digits. Machine learning has been doing quiet work like this since the nineteen-nineties: reading cheques, scoring loan applications, ranking search results, choosing what to show you. It is everywhere, but mostly out of sight.

What is new is that it now does things people can see and that were thought impossible. Voice assistants answer spoken requests reasonably well, from switching on a light to booking a haircut, and speech recognition now matches people at some tasks. When you ask your voice assistant for tomorrow's weather and it understands you in a noisy room, you are hearing this progress. Recognising objects in photos went from hard to routine: on a standard test, the best systems in 2010 missed about twenty-eight per cent of the time, even when allowed five guesses; by 2017 they missed about two per cent. The same methods now identify birdsong and help diagnose skin cancer.

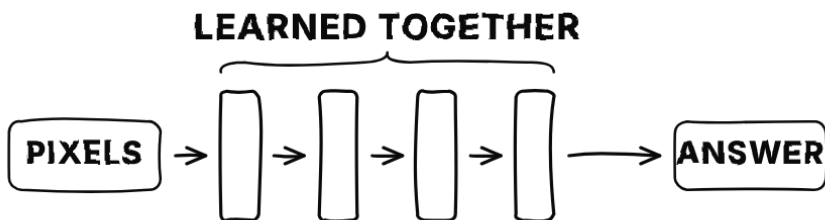


Error on ImageNet

Games have long been a yardstick. A backgammon program learned by reinforcement decades ago; Deep Blue beat Kasparov at chess with special hardware and brute search; Go, with far more possible positions, fell to AlphaGo, which combined deep learning with sampling the tree of moves; and a program called Libratus beat top players at poker, a game in which you cannot see your opponents' cards. Self-driving cars have made real progress, though full autonomy is still out of reach; there, deep learning mostly handles seeing, while engineers tune the rest by hand. Robotics, logistics, biology, particle physics and astronomy all owe recent advances to these tools.

What about the fear of machines that become conscious and turn on us? The authors are calm about it. Today's systems are built for specific goals, and behind any appearance of general intelligence lie rules, rules of thumb and statistical models; there are no tools for a machine that improves and redesigns itself to solve any task. The worries they find more pressing are nearer at hand: routine jobs being automated, as farm robots will automate harvesting, with large effects on people who depend on such work; and careless models making biased decisions about loans, jobs or bail. The same care applies to the recommendation feed on your phone, which makes small automated decisions about what millions of people see every day.

So what, in the end, is deep learning? It is machine learning with models built from many layers of transformations, all learned together from the data. Before it, experts spent years designing by hand the steps that turn raw pixels or sound into useful features, and only then handed those features to a simple model. Deep learning trains the whole chain at once, end to end, from the raw input to the answer, and the filters it finds for itself beat the ones people designed. That is why it shines on raw sound, raw pixels and sentences of any length, and why it has dissolved the walls between vision, speech, language and medicine, giving them one shared set of tools.



Hand-made versus end to end

It also reflects a change in style. With plenty of data, the field relies less on simplifying assumptions and more on flexible models that fit the data closely, much as physicists moved from hand-solved approximations to computer simulations: more accurate, though often harder to interpret. It accepts solutions that are good rather than provably best, and tries things before proving them. And it shares: tools, models and trained networks are released openly, which is also why this book is free.

That closes the introduction. Across these parts, the story has run from programs written as rules to programs that learn from examples, through the many shapes learning can take, to the data, chips and tools that let deep networks finally work.

## Remember

Machine learning has long worked out of sight, in fraud checks, postal sorting and search, and deep learning has now brought visible successes in speech, images and games. The authors see the real risks not in conscious machines but in lost jobs and biased automated decisions. Deep learning means many layers of transformations learned together, end to end, from raw data to answer. It replaced hand-designed features and gave fields such as vision, speech and language one shared set of tools. Its progress rests on plenty of data, cheap computing and graphics cards, and open frameworks that anyone can use.

---

## Questions

1. What chiefly sets deep learning apart from earlier machine learning, according to this part?
  - a. It uses no data at all
  - b. Its many layers are learned together, end to end, instead of hand-designed features feeding a simple model
  - c. It works only on games
  - d. It always finds the provably best solution
2. What do the authors consider the more pressing risk of AI today?
  - a. Machines becoming conscious and turning on their makers
  - b. Computers running out of memory

- c. Routine jobs being automated and careless models making biased decisions
  - d. Games becoming too easy for computers
- 3. Why was poker a different kind of challenge from chess for a computer?
  - a. Poker has fewer possible positions than backgammon
  - b. Poker has no rules
  - c. Chess cannot be learned by reinforcement
  - d. In poker you cannot see your opponents' cards, so the game is only partly observed
- 4. Why has deep learning broken down the walls between fields such as vision, speech and language?
  - a. Because end-to-end training replaces much of the field-specific preparation, so the same tools work across them
  - b. Because all these fields now use the same datasets
  - c. Because vision and speech are the same problem
  - d. Because frameworks forbid specialised methods

## Answers

- 1. **b)** Before, experts designed the feature steps by hand; deep learning trains the whole chain from raw input to answer at once.
- 2. **c)** They see no tools for self-improving general intelligence, but real effects from automation and from biased automated decisions about loans, jobs or bail.
- 3. **d)** Chess shows the whole board, but in poker part of the situation is hidden, which makes reasoning much harder.

**4. a)** Once the features are learned from raw data rather than designed by experts in each field, one shared set of tools serves them all.

## Worked examples

**How much better object recognition became** On the ImageNet test, the best systems in 2010 had a top-five error rate of 28 per cent; by 2017 it was 2.25 per cent. Out of 10,000 test images, how many did each get wrong, and how many times fewer mistakes is that?

The error rate is the share of examples the model gets wrong.

In 2010: 28 per cent of 10,000 is 2,800 images wrong.

In 2017: 2.25 per cent of 10,000 is 225 images wrong.

2,800 divided by 225 is about 12.4.

**Answer:** 2,800 mistakes against 225: about twelve times fewer

## Practice

**1.** A system with a top-five error rate of 2.25 per cent is run on 40,000 photographs. About how many does it get wrong, and how many would a 2010 system with 28 per cent error have got wrong?

**Hint:** Multiply the number of photographs by each error rate written as a fraction.

**Answer:** 900 for the 2017 system ( $0.0225 \times 40,000$ ) and 11,200 for the 2010 system ( $0.28 \times 40,000$ ).

**2.** Explain in your own words the difference between the old way of recognising objects in images, with hand-designed feature extractors, and the deep learning way. Why did the new way win?

**Hint:** Ask who decides how raw pixels are turned into features in each case.

**Answer:** In the old way, experts designed by hand the steps that turn pixels into features, such as edge detectors, and then a simple model used those features. In deep learning the whole chain, features included, is learned from data end to end. It won because an algorithm testing millions of choices automatically finds better filters than human ingenuity alone.