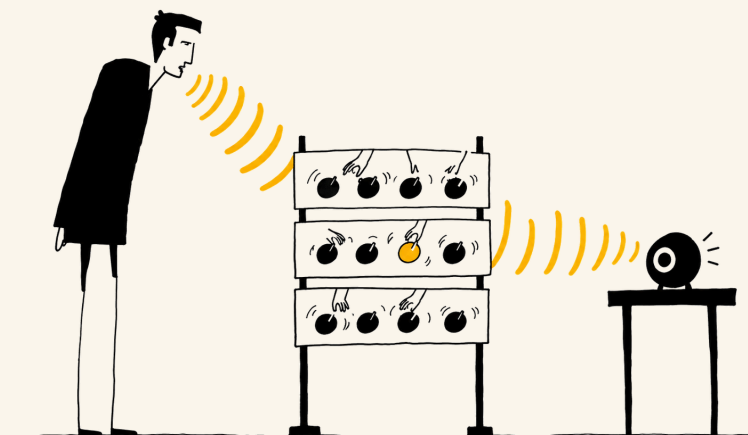


Dive into Deep Learning



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

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

Contents

A shop that never learns

Programming with examples

Good data, bad data

Practice papers and the real exam

Guessing how much

Which one is it?

Tags, rankings and your feed

When order matters

Learning without answers

Learning by doing

Old roots

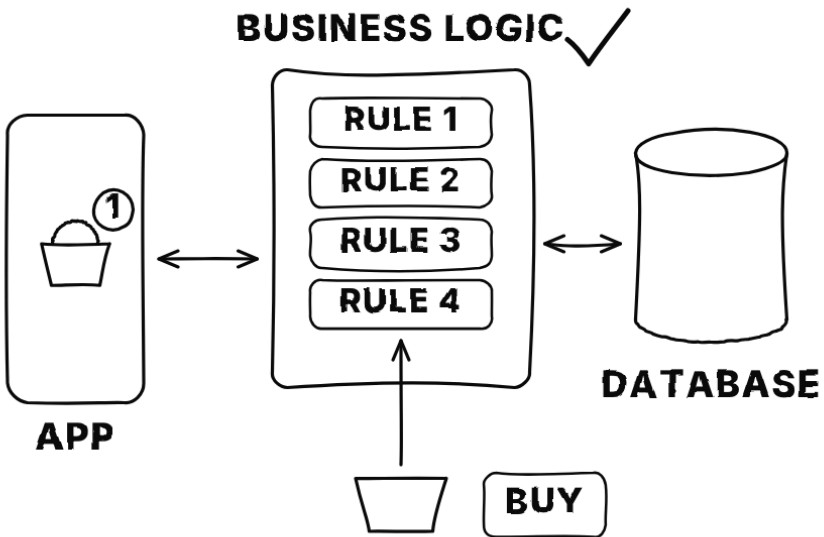
The great acceleration

Everywhere, mostly unseen

What makes it deep

A shop that never learns

Open a shopping app, tap "add to basket" on a pair of socks, and watch the little number on the basket jump to one. Do it again tomorrow, or a thousand times, and it jumps in exactly the same way. Someone wrote a rule for that tap long ago, and the rule does not change because you used it.



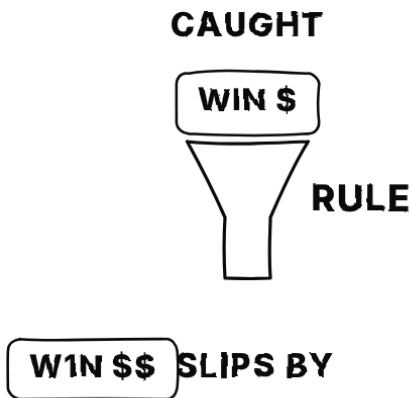
A shop built from rules

Most programs you meet in a day were built like that, as rules written by people. The book imagines a team at a whiteboard planning an online shop. Customers will see it in a browser or a phone app; a

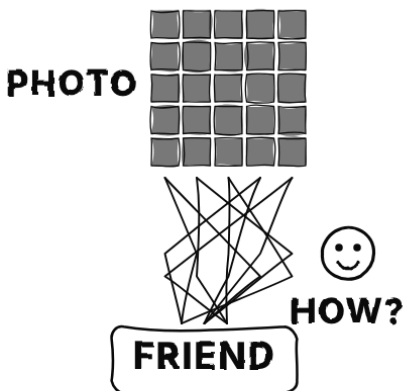
database will remember every customer and every purchase; and in the middle sits what the authors call the business logic, the brains of the shop, a set of rules saying what to do in every situation. The team then hunts for awkward cases, such as someone pressing "buy" with an empty basket, and writes a rule for each one. After some testing it works, and the shop can open before a single real customer arrives. The authors admire this, and they add a blunt piece of advice: if you can write rules that are right every single time, you have no need for machine learning.

Now give the same team harder jobs. Predict tomorrow's weather from satellite pictures and past weeks of measurements. Answer a question typed in everyday words. Find every person in a photo and draw around each one. Suggest things a shopper will love but would never have found by browsing. That last job is what your recommendation feed attempts every time you open it. Even the best programmers would struggle to write the rules for any of these.

MOVING TARGET



TOO TANGLED

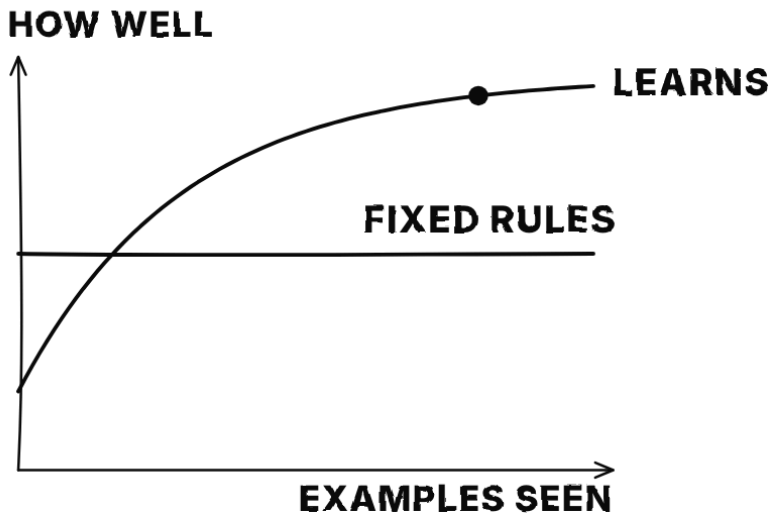


Two reasons rules fail

There are two reasons. Sometimes the right answer keeps moving, so fixed rules go stale and a good solution has to adjust as the world changes. A spam filter lives with this: as soon as it catches one kind of junk mail, the senders change their wording. At other times the link between what goes in and what should come out is far too tangled to write down. You recognise a friend's face in a crowd in an instant, yet you could never list the steps your brain took to do it.

Machine learning is the study of methods that learn from experience, usually from many recorded examples, and that get better as they see more. The shop's rules stay the same however many customers pass through, until a programmer decides to rewrite them. A learning system improves from the very examples it sees. When a voice

assistant gets better at understanding how people really speak, no one wrote a rule for each voice; it learned from large amounts of recorded speech. The book is about one powerful family of these methods, called deep learning, which lies behind much recent progress in computer vision, understanding language, medicine and the study of genes. The chatbot you type to is one result: nobody wrote its answers as rules; it learned the patterns of language from a huge amount of text.



Fixed rules versus learning

Remember

Ordinary programs are rules written by people, and they behave the same however much they are used. When people can foresee every case and get it right every time, rules are enough. Some tasks resist rules, because the answer keeps changing or because the path to it is too tangled to write down. Machine learning is about methods that improve with experience, and deep learning is the family of those methods this book teaches.

Questions

1. Why does the basket number jump the same way however often you tap it?
 - a. The app is still collecting examples to learn from
 - b. It follows a rule written in advance that does not change with use
 - c. The phone forgets each tap straight away
 - d. Shopping habits change too fast to learn
2. When does the book say you have no need for machine learning?
 - a. When the task involves a phone app
 - b. When there is not much data yet
 - c. When you can write rules that are right every single time
 - d. When customers have not arrived yet

3. A spam filter keeps missing new junk mail. Which of the two reasons for rules failing does this show?
 - a. The right answer keeps moving as senders change their wording
 - b. The link from mail to answer can never be learned
 - c. Programmers forgot to test the empty-basket case
 - d. The filter has seen too many examples
4. What makes a learning system different from the shop's rules?
 - a. It never makes mistakes
 - b. It is written by better programmers
 - c. It needs no examples at all
 - d. It improves from the examples it sees

Answers

1. **b)** Someone wrote a rule for that tap long ago, and the rule does not change because you used it.
2. **c)** If you can write rules that are right every single time, you have no need for machine learning.
3. **a)** Sometimes the right answer keeps moving: as soon as the filter catches one kind of junk mail, the senders change their wording.
4. **d)** The shop's rules stay the same until a programmer rewrites them, while a learning system improves from the very examples it sees.

Practice

1. For each of the four harder jobs (tomorrow's weather, answering a typed question, drawing around every person in a photo, suggesting things a shopper would never find), say which reason makes rules

hard: the answer keeps changing, or the path to it is too tangled to write down.

Hint: Ask whether the right answer drifts over time, or whether you simply could not list the steps.

Answer: Weather and shoppers' tastes keep changing, so fixed rules go stale, and weather is tangled as well. Answering everyday questions and finding people in a photo are the tangled kind: we do them easily but cannot list the steps. Several jobs have both problems.

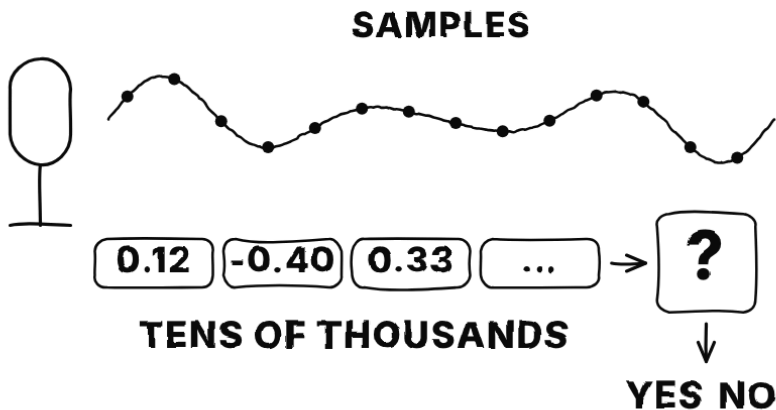
2. Explain in your own words why a rule is fine for a customer pressing buy with an empty basket.

Hint: Can the team see this case coming and write the right response once?

Answer: The team can foresee this case before any customer arrives and write a response that is right every time, such as a message saying the basket is empty. When rules can be right every time, machine learning is not needed.

Programming with examples

Say "Hey Siri" or "OK Google" across a noisy room and watch your phone wake. Out of the television, the kettle and everyone talking, it caught one phrase. Now picture writing that program yourself, with only a computer and a blank screen. Where would you even start?



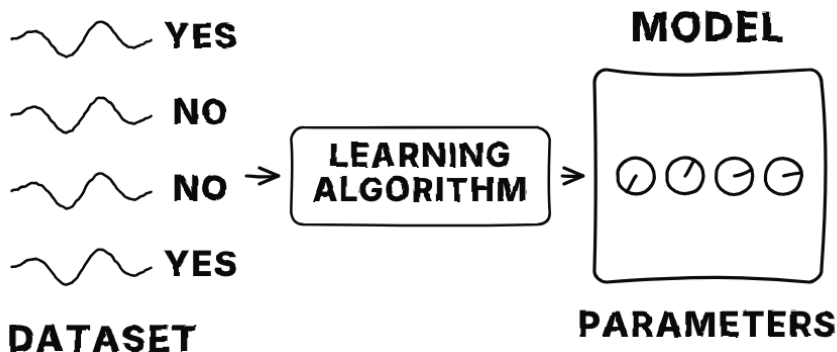
Sound in, yes or no out

The authors tell a small scene, made up to make the point. On the way to buy coffee, one of them wakes a phone with "Hey Siri" and asks for directions to a coffee shop. The phone writes out his words, understands that he wants directions, opens the maps, and shows

several routes with a predicted travel time beside each. In a few seconds, several separate learned programs did their work. When you ask a voice assistant to set a timer, the same chain runs: one program listens for the wake phrase, another writes down your words, another works out what you meant.

Take just the first job, noticing the wake word. The microphone turns sound into a long stream of numbers, tens of thousands every second. From that stream the program must answer only yes or no: was the phrase said? Nobody knows a rule that does this reliably, and the book admits that its authors do not know one either. That is exactly why machine learning is used.

The trick is this: we cannot explain how to recognise "Alexa", but we can recognise it ourselves. So people listen to a great many short recordings and mark each one yes or no. That collection is called a dataset, and the marks are labels. Instead of writing a recogniser, we write a flexible program with many adjustable settings, called parameters, and let the dataset decide where they should be. The book pictures the parameters as knobs on a machine. With the knobs set and left alone, the program is called a model, and the program that uses the data to choose the knob settings is the learning algorithm.



Labels, knobs and a learner

One kind of machine, with its knobs turned one way, says yes to "Alexa"; turned another way, it could say yes only to "Apricot", because the two jobs are alike. Turning photos into captions or translating English into Chinese is a very different job, and may need a different kind of machine.

Try it yourself

Ask a friend to say a word in a whisper, a mumble and a shout from the next room. You will recognise it every time, yet you will not manage to write down what makes it that word and not a similar one. That gap, between being able to do something and being able to state the rule, is why we learn from examples.

With the knobs set at random, the model recognises nothing. Training is how the right setting is found, and the book draws it as a loop. Start with a model that does nothing useful, show it some labelled recordings, nudge the knobs so it does a little better on them, and repeat with more examples until it is very good. The authors call this programming with data. A cat detector can be "programmed" the same way, by showing it many pictures of cats and dogs, until it gives a strong yes for cats, a strong no for dogs, and something in between when unsure. Your spam filter learned like this: it was shown many messages marked spam or not, and its knobs settled wherever they separate the two. Deep learning, the subject of the book, is one of several popular ways of doing this.

Remember

A few seconds with a phone can involve several learned programs at once. Nobody can write rules to spot a wake word in raw sound, but people can recognise it, so they label many examples. A flexible program with adjustable knobs, its parameters, is then trained on those examples, round after round, until it does the job. This is programming with data, and deep learning is one way of doing it.

Questions

1. Why can people help a computer learn a wake word they could never write rules for?
 - a. Because they can recognise the word themselves and mark recordings yes or no
 - b. Because they can write down what the sound looks like as numbers
 - c. Because the phone already knows the word when it is made
 - d. Because the word is always said the same way
2. What do the knobs in the book's picture stand for?
 - a. The labels on the recordings
 - b. The adjustable settings, the parameters, that decide how the program behaves
 - c. The buttons the user presses on the phone
 - d. The number of recordings in the dataset

3. Why might the same kind of machine do both "Alexa" and "Apricot" but not turning photos into captions?
 - a. Captions need a bigger dataset than any machine can use
 - b. Photos cannot be labelled
 - c. The two wake words are alike jobs, while captioning photos is a very different job
 - d. "Apricot" has more letters than "Alexa"
4. What happens at each round of training?
 - a. All the knobs are set at random again
 - b. A programmer writes one more rule
 - c. The dataset is thrown away and collected again
 - d. The knobs are nudged so the model does a little better on some labelled examples

Answers

1. **a)** We cannot explain how to recognise the word, but we can recognise it ourselves, so we mark many recordings for the program to learn from.
2. **b)** The book pictures the parameters, the program's adjustable settings, as knobs on a machine.
3. **c)** The two wake word jobs are alike, so another turn of the same knobs will do; captioning photos is a very different job and may need a different machine.
4. **d)** Show the model some labelled recordings, nudge the knobs so it does a little better on them, and repeat.

Practice

1. Put these training steps in order: repeat until the model is good; start with a model that does nothing useful; nudge the knobs; show it some labelled examples. Why is the first model useless?

Hint: What has a model learned before it has seen any examples?

Answer: Start with a model that does nothing useful, show it some labelled examples, nudge the knobs so it does better on them, and repeat until it is good. The first model is useless because its knobs are set at random and it has learned nothing yet.

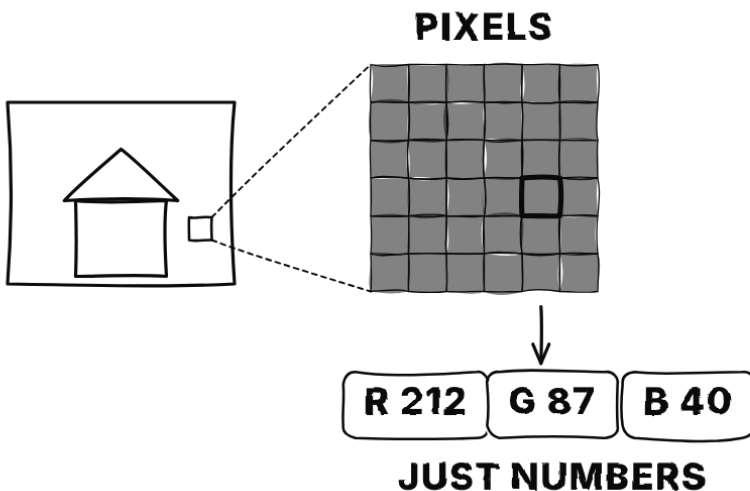
2. Explain in your own words why the authors call training "programming with data".

Hint: Who decides how the finished program behaves?

Answer: Nobody writes the program's behaviour rule by rule. It is set by showing the program many labelled examples and letting the knobs settle where the program does well, so the examples do the job that hand-written rules would have done.

Good data, bad data

Zoom into a photo on your phone as far as it goes, and the picture breaks into tiny coloured squares. Each square is just three brightness readings, one for red, one for green, one for blue. To a learning program your holiday snapshot is nothing more than a long list of such numbers. That is the raw material every learning machine works with.



A photo is only numbers

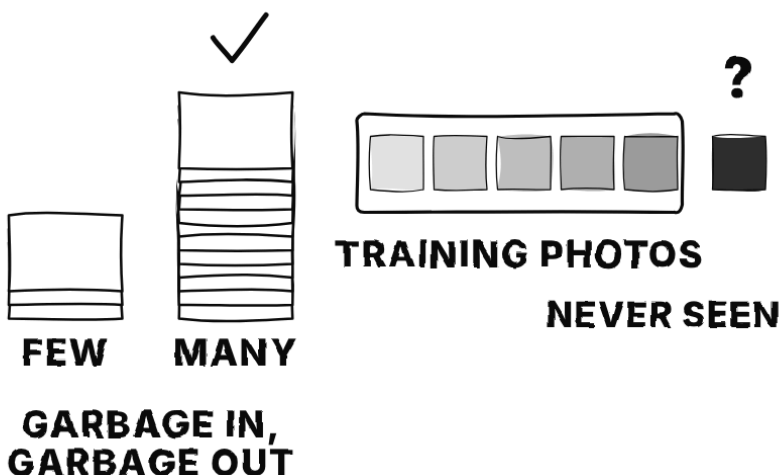
The book names four parts that every machine learning problem has, whatever its kind. First, the data the machine learns from. Second, the

model, the knob-covered program that turns that data into an answer. Third, a way of scoring how well or how badly the model is doing. Fourth, a method for turning the knobs so that the score improves. The next parts deal with the last three; here the subject is the data.

Data is a pile of examples, and each example has to be turned into numbers before a machine can use it. A photo becomes its pixel readings, and even a small square photo already turns into well over a hundred thousand numbers. A hospital record becomes a patient's age, pulse, other illnesses, medicines and recent operations, all written as numbers, and the thing to predict might be whether the patient survives the next month. Your spam filter sees each message the same way, as a set of numbers describing it, with "spam" or "not spam" as the answer it must give.

It is easiest when every example comes as the same number of readings, like photos all taken by the same microscope. Real life is messier. Pictures found on the internet come in every size and shape; you can trim them all to one size, but whatever you cut off is lost. Text is worse: one shopping review says only "it stinks!", another goes on for pages. The book counts it as one of deep learning's great strengths that it copes with examples of different lengths far more gracefully than older methods. That is why the chatbot you type to accepts a three-word question as happily as a three-page one.

More data usually makes everything easier. With more examples, a more powerful model can be trained and fewer guesses have to be built in beforehand. The book says the switch from small collections of examples to enormous ones is one of the main reasons deep learning took off; many of its best models simply do not work without huge amounts of data.



More data, the right data

Yet a mountain of data is useless if it is the wrong data. If the examples are full of mistakes, or describe things unrelated to what we want to predict, learning fails: garbage in, garbage out. And where machines help decide about people, in policing, hiring or lending, bad data does real harm. One danger is that some people are missing from the examples altogether, like a skin cancer detector that was never shown dark skin. Another is that the examples carry old prejudice: a program trained on a firm's past hiring decisions can learn to repeat their unfairness automatically. The authors stress that this can happen without anyone intending it, or even noticing.

Remember

Every learning problem has four parts: data, a model, a score of how well it does, and a method for turning its knobs. Data must become numbers, and a single small photo already becomes over a hundred thousand of them. Examples that come in different sizes, like reviews and internet photos, are awkward, and deep learning handles them comparatively well. More data helps, but only the right data: mistakes, missing groups or old prejudice in the examples turn into failures and unfairness in the machine.

Questions

1. Why must a photo be turned into numbers before a machine can learn from it?
 - a. Because a learning machine works only with numbers, such as brightness readings for each tiny square
 - b. Because numbers take less space than pictures
 - c. Because labels can only be attached to numbers
 - d. Because the photo would otherwise be too large to crop
2. Why are shopping reviews awkward data?
 - a. They are written by too few people
 - b. They come in very different lengths, from two words to pages
 - c. They contain no useful information
 - d. They cannot be stored on a computer

3. A program trained on a firm's past hiring decisions treats some applicants unfairly. What is the likely cause?
 - a. It was given too much data
 - b. Its knobs were never turned
 - c. The examples carried old prejudice, which it learned to repeat
 - d. The applications were too short
4. Why did the move to enormous collections of examples matter so much?
 - a. Because large data needs no scoring
 - b. Because older methods could not read text
 - c. Because big data removes all prejudice
 - d. Because many of the best deep learning models simply do not work without huge amounts of data

Answers

1. **a)** To a learning program a photo is nothing more than a long list of brightness readings; every example has to become numbers.
2. **b)** One review says only "it stinks!", another goes on for pages, so they do not come as the same number of readings.
3. **c)** If the examples carry old prejudice, a program trained on them can learn to repeat that unfairness automatically.
4. **d)** The switch from small collections to enormous ones is one of the main reasons deep learning took off; many of its best models need huge amounts of data.

Practice

1. A skin cancer detector works very well in its makers' tests but often fails in a clinic serving many patients with dark skin. Explain what probably went wrong and how it could be avoided.

Hint: Think about who was in the examples the detector learned from.

Answer: People with dark skin were probably missing, or nearly missing, from the training examples, so the detector never learned what the illness looks like on dark skin. The fix is to gather examples that cover everyone the detector will be used on.

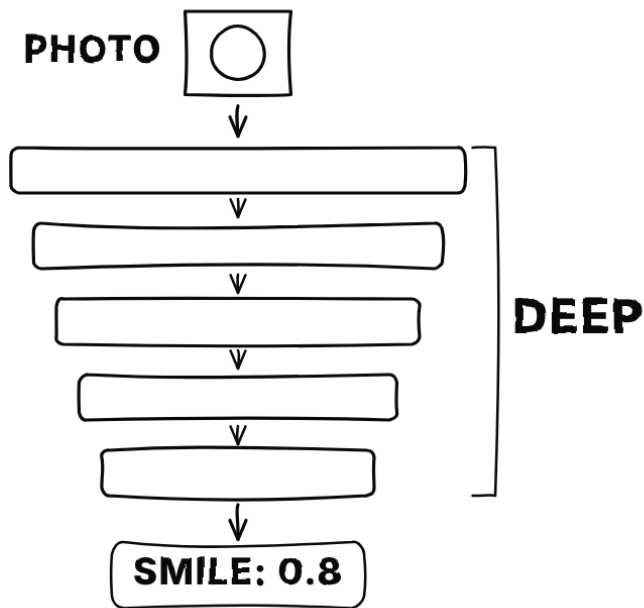
2. Name the four parts every learning problem has, and say in your own words what each is for.

Hint: What it learns from, what gives answers, what scores, what adjusts.

Answer: The data is what the machine learns from; the model is the program with knobs that gives the answers; the score says how well or badly it is doing; and the method for turning the knobs changes them so the score improves.

Practice papers and the real exam

A student revises with last year's practice papers until she gets every question right. Then the real exam comes, with questions she has never seen, and she struggles. She had memorised those papers instead of learning the subject. Learning machines can fall into exactly the same trap, and this part is about how we notice.

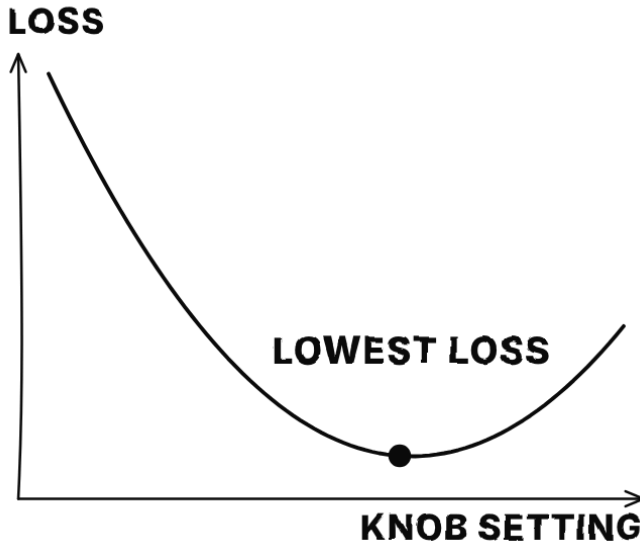


Deep means many steps

Start with the model. It takes in data of one kind and gives out a prediction, often of another kind: a photo goes in and out comes how much the person is smiling, or a set of sensor readings goes in and out

comes how unusual they look. Simple models are fine for simple jobs, but the jobs in this book need something stronger. What makes deep learning different is its powerful models, built as many steps that each reshape the data a little and pass it on to the next, stacked one on top of another. That tall stack of steps is where the word "deep" comes from.

Next, how do we know the model has got better? People can argue about whether a change is an improvement, so learning machines need a firm score. That score is called the objective function, and by custom it is set up so that lower is better, which is why it is also called the loss. For predicting a number, the usual loss grows quickly the further the guess is from the truth; for sorting things into kinds, what counts in the end is how often the model is wrong. That second score is awkward to improve directly, so in practice a smoother stand-in score is often used during training. A spam filter is finally judged by how much mail it sorts wrongly, even if it learns from a smoother score along the way.

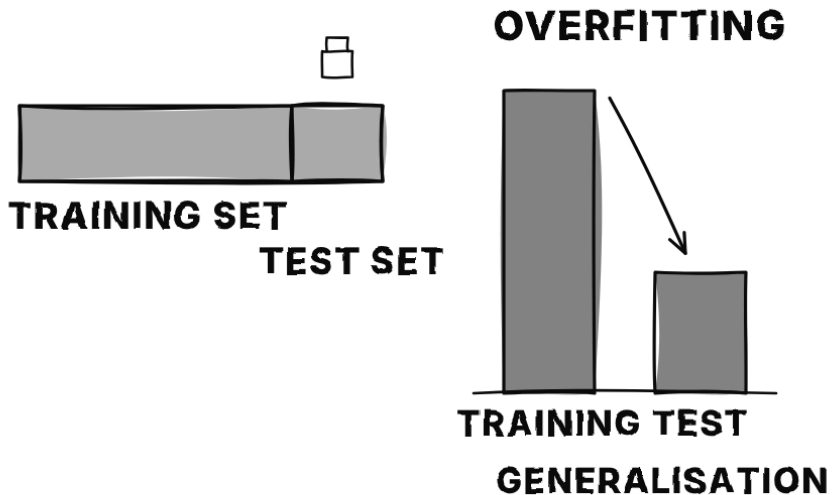


Walking downhill on the loss

Then we need a way to turn the knobs so the loss goes down, and nearly all of deep learning uses one method, called gradient descent. At each step it checks, for every knob, whether turning it a tiny bit would make the loss on the training examples go up or down. Then it turns each knob a little in the direction that lowers the loss, and repeats. Picture walking down a hillside in thick fog: you cannot see the valley, but you can feel which way the ground slopes under your feet, take a small step downhill, and feel again.

Here the student returns. The knobs are tuned using one pile of examples, the training set, and checked on another pile kept aside and never used for tuning, the test set. A good score on the training set is like a good score on the practice papers: encouraging, but no promise.

Doing well on new, unseen examples is called generalisation, and a model that shines on its training examples but fails on new ones is said to be overfitting. A recommendation feed that perfectly matches what you clicked last month but suggests nothing you want next is the kind of failure a test set is meant to catch.



Training set and test set

Remember

A model turns data into predictions, and deep learning's models are tall stacks of steps, which is why they are called deep. The loss is a firm score of how badly a model is doing, and lower is better. Gradient descent lowers the loss by feeling, knob by knob, which way is downhill and stepping a little that way, again and again. Success is checked on a test set the model never trained on, because a model that only memorised its training examples is overfitting.

Questions

1. Where does the word "deep" in deep learning come from?
 - a. From the huge amounts of data it needs
 - b. From its models being tall stacks of steps, each reshaping the data and passing it on
 - c. From how hard its ideas are to understand
 - d. From how low its loss can go
2. What does gradient descent do at each step?
 - a. Sets all the knobs to new random values
 - b. Checks the test set and stops if it is good
 - c. Feels, for every knob, which way lowers the loss, and turns each knob a little that way
 - d. Adds more examples to the training set

3. Why is a model checked on a test set it never trained on?
- a. Because a good score on familiar examples does not show it will do well on new ones
 - b. Because the training set is always too small
 - c. Because the test set has a lower loss
 - d. Because gradient descent only works on the test set

Answers

1. **b)** Deep learning's models are built as many steps stacked one on top of another, and that tall stack is where the word comes from.
2. **c)** It checks for every knob whether a tiny turn would raise or lower the loss, then turns each a little in the direction that lowers it.
3. **a)** A good score on the training set is like a good score on practice papers: encouraging, but no promise for new questions.

Practice

1. Explain how walking down a foggy hillside is like gradient descent. What stands for the height of the ground, and what for your position?

Hint: What is gradient descent trying to make small?

Answer: The height of the ground stands for the loss, and your position stands for the settings of the knobs. You cannot see the bottom, but you can feel which way is downhill where you stand and take a small step that way, just as gradient descent checks which way each knob lowers the loss and turns it a little, again and again.

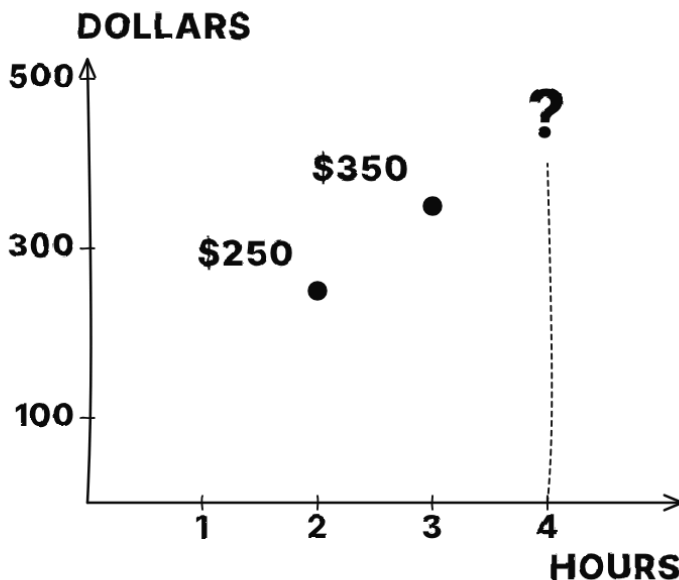
2. A model is right on 99 of 100 training examples but on only 60 of 100 test examples. What has happened, and what does the student story say about it?

Hint: Compare the practice papers and the real exam.

Answer: The model is overfitting: it has, in effect, memorised its training examples and does not generalise to new ones, like the student who memorised the practice papers and then struggled with new questions in the real exam.

Guessing how much

A plumber spends three hours clearing your blocked drains and sends a bill for three hundred and fifty dollars. Your friend uses the same plumber for two hours and pays two hundred and fifty. A neighbour now asks what a four-hour job would cost. You can probably answer without any maths lesson, and the way you do it is the heart of this part.



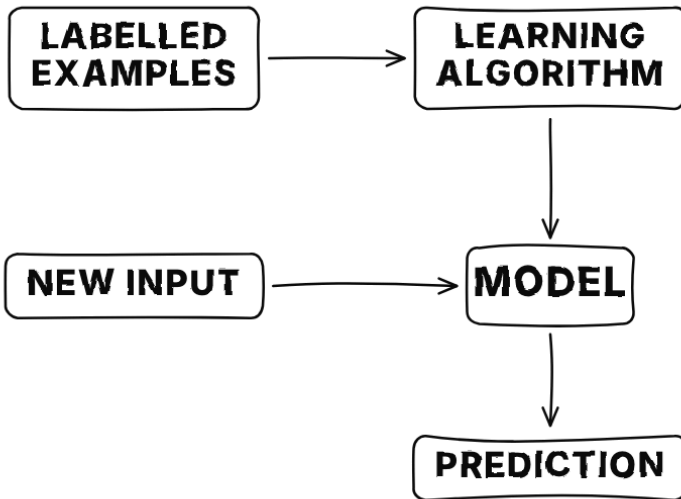
Two plumber's bills

First, a broader view. The wake word detector is only one of many problems machine learning can take on, and the book now walks

through the main kinds, starting with the most common. In supervised learning, we are given examples where we know both what goes in, the features, and the right answer, the label. The job is to build a model that, shown the features of a new case, predicts its label. It is called supervised because we act as supervisors, handing the model labelled examples to learn its knob settings from. More exactly, the model is estimating how likely each answer is, given what it can see.

Most of the machine learning that works well in business is of this kind, because so many tasks fit the pattern: guess something unknown from what is known. Is there cancer in this scan or not? What is the French for this English sentence? What will this company's shares be worth next month, judging by this month's accounts? Your spam filter fits the same mould: it is shown a message and guesses spam or not spam. The pattern is simple, but it takes many forms; texts of any length, for instance, need different models from inputs that always come at one size.

The book sketches the process, and draws it as a picture with arrows. Gather a big pile of examples, pick some at random, and get the right answer for each. Sometimes the answers are already on record, such as whether a patient lived through the following year; sometimes people must be hired to label things by hand, such as sorting photos into categories. Those labelled examples are the training set. They go into a learning method, and out comes a trained model; then new cases, never seen before, go into the model, and its outputs are taken as its predictions.



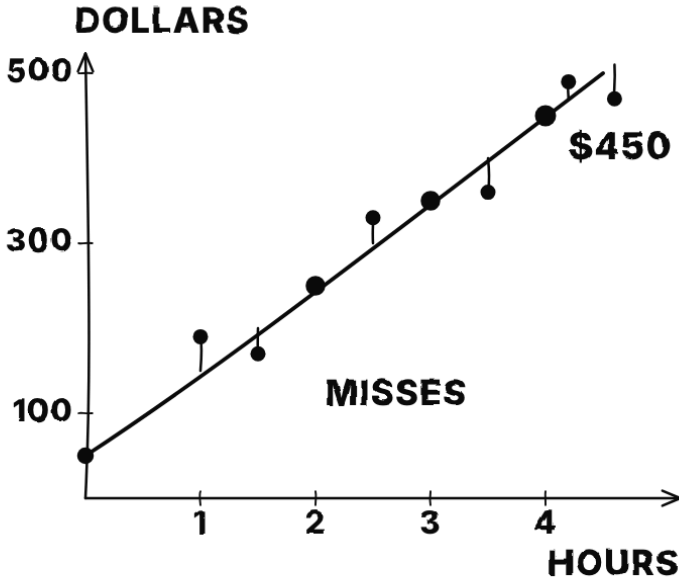
From labelled examples to predictions

The simplest kind of supervised learning to grasp is regression. Imagine a table of house sales, with one row for each house and one column for each fact about it: floor area, bedrooms, bathrooms, and minutes on foot to the town centre. Each row is one example, a fixed list of facts about one house. A small flat in New York might be a tiny floor area, one bedroom, one bathroom and an hour's walk from the centre; a family house in Pittsburgh, a much bigger area, four bedrooms, three bathrooms and ten minutes' walk. Most classic learning methods need every example to come as such a list of the same length.

What makes a problem regression is the kind of answer wanted: a number, such as the price a house sold for. If you are buying a home,

a model trained on past sales could estimate a fair price from the facts about a house. A handy rule is that any question asking "how much?" or "how many?" is probably regression. How long will this operation take, how much rain will fall here in the next six hours, how many days will this patient stay in hospital? When a streaming service guesses how many stars you would give a film, that is regression too, and in 2009 a team that did it well enough won a million-dollar prize from Netflix.

Now back to the plumber. You probably assumed that more hours cost more, and that there is a fixed call-out charge plus a rate for each hour. With that assumption, the two bills give the answer at once: the extra hour cost a hundred dollars, so the rate is a hundred an hour, and fifty dollars is just for turning up. If you followed that, you already understand the idea behind linear regression, the plainest kind.



A call-out charge plus a rate

Here the two bills fit the rule exactly. In real data they rarely do, because things you have not measured also push the answers around: a harder blockage, a longer drive, a bad day. Then the model cannot match every example, so it tries to keep its guesses as close as possible to the real answers overall. The usual way of measuring "close" punishes big misses much more than small ones, and the book will mostly use it. The authors add that this choice is the natural one when the stray variations behave like ordinary random noise.

Remember

Supervised learning means learning to predict a label from features, using examples where the right answer is given. Most successful real-world machine learning works this way, from medical scans to translation. Regression is supervised learning where the answer is a number: any "how much?" or "how many?" question. The plumber's two bills reveal a call-out charge plus an hourly rate, which is the idea of linear regression. When real data does not fit exactly, the model keeps its guesses as close as it can, punishing big misses most.

Questions

1. Which of these questions is a regression problem?
 - a. Is this message spam or not?
 - b. Is there cancer in this scan?
 - c. How many days will this patient stay in hospital?
 - d. What is the French for this sentence?
2. Why is this kind of learning called supervised?
 - a. Because we hand the model labelled examples to learn its knob settings from
 - b. Because someone checks each prediction by hand
 - c. Because it is used mostly in business
 - d. Because the model can only run while watched

3. Why can a model trained on real house prices usually not match every sale exactly?
 - a. Because houses have too few features
 - b. Because things not in the table, such as a buyer's hurry, also push prices around
 - c. Because regression only works for plumbers
 - d. Because the training set is never labelled
4. How does the usual measure of closeness treat a large miss compared with a small one?
 - a. It ignores large misses
 - b. It counts all misses the same
 - c. It counts only the largest miss
 - d. It punishes the large miss much more

Answers

1. **c)** Regression is supervised learning where the answer is a number; any "how much?" or "how many?" question is probably regression.
2. **a)** We act as supervisors, handing the model labelled examples to learn its knob settings from.
3. **b)** In real data, things you have not measured also push the answers around, so the model keeps its guesses as close as it can overall.
4. **d)** The usual way of measuring close punishes big misses much more than small ones.

Worked examples

The plumber's prices A plumber charged 350 dollars for a 3-hour job and 250 dollars for a 2-hour job. Assuming a fixed charge for

turning up plus a rate for each hour, what are the charge and the rate, and what would a 4-hour job cost?

The two jobs differ by one hour and by 100 dollars, so each hour costs 100 dollars.

Three hours cost 300 dollars of the 350, so 50 dollars is the fixed charge for turning up.

Check with the other bill: 50 plus 2 hours at 100 is 250, as charged.

A 4-hour job would be 50 plus 4 hours at 100.

Answer: 100 dollars an hour plus 50 dollars to turn up; a 4-hour job costs 450 dollars

Practice

1. Sort these into regression or not: how much rain will fall in the next six hours; is this photo a cat or a dog; how many hours will this operation take; what price will this house sell for.

Hint: Is the answer a number, a how much or how many?

Answer: Rain, operation time and house price ask for numbers, so they are regression. Cat or dog asks for a category, so it is not regression.

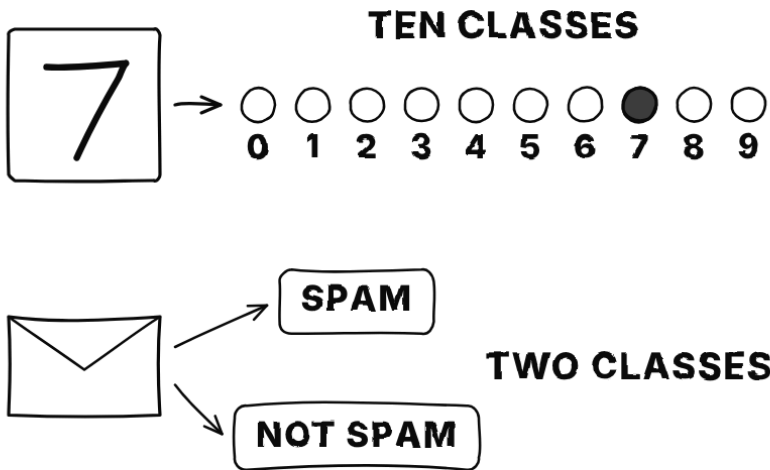
2. A second plumber charged 200 dollars for 1 hour and 400 dollars for 3 hours. Assuming the same kind of pricing, what are this plumber's fixed charge and hourly rate?

Hint: Two extra hours added 200 dollars.

Answer: Two hours added 200 dollars, so the rate is 100 dollars an hour; one hour's 100 leaves 100 dollars of the first bill, so the fixed charge is 100 dollars.

Which one is it?

Picture a beautiful pale mushroom growing in your garden. You photograph it with an app that says it is eighty percent sure the mushroom is harmless. Would you fry it for dinner? Hold on to your answer; the book comes back to this mushroom.

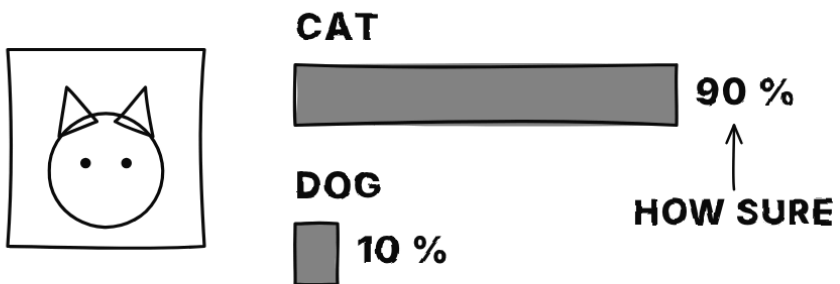


Which class is it?

Regression answers "how much?", but many questions are "which one?". Take a banking app that reads a cheque from a photo. Once the picture is cut into small patches, one for each handwritten character, the job left is to decide which character, out of a known set, each

patch shows. Problems like this are called classification, and they need somewhat different tools from regression, though much carries over.

In classification, the model looks at the features of an example, such as the pixels of an image, and says which category, or class, it belongs to, out of a fixed set of options. For handwritten digits there are ten classes, nought to nine. The simplest case has only two classes, such as cat or dog, and is called binary classification. Your spam filter is binary classification: every message is either spam or not. Where regression produces a number, classification produces a choice of class. When there are more than two classes, as with all the digits and letters of handwriting, it is called multiclass classification.



A probability for each class

A model that can only say a flat "cat" or "dog" turns out to be hard to train, so instead it usually gives a probability for each class. Shown a picture, it might say ninety percent cat, which we can read as "ninety percent sure". How large that probability is gives a sense of how unsure the model is. Just as regression is usually trained by keeping squared misses small, classification has its own usual score of how wrong the model is, with the odd name cross-entropy, which the book will explain much later.

Now back to the mushroom. Suppose the app says there is a twenty percent chance it is a death cap, and so eighty percent that it is not. The most likely answer is "safe", yet only a fool would eat it. The pleasure of one good dinner is not worth a one-in-five chance of dying. To decide what to do, it is not enough to know which answer is most likely; you have to weigh each choice by how likely each outcome is and how good or how terrible it would be. Eating risks a catastrophe; throwing the mushroom away costs at most a lost dinner. And the caution was right: the mushroom in the book's picture really is a death cap.

 **SAFE 80 %**

 **DEATH CAP 20 %**



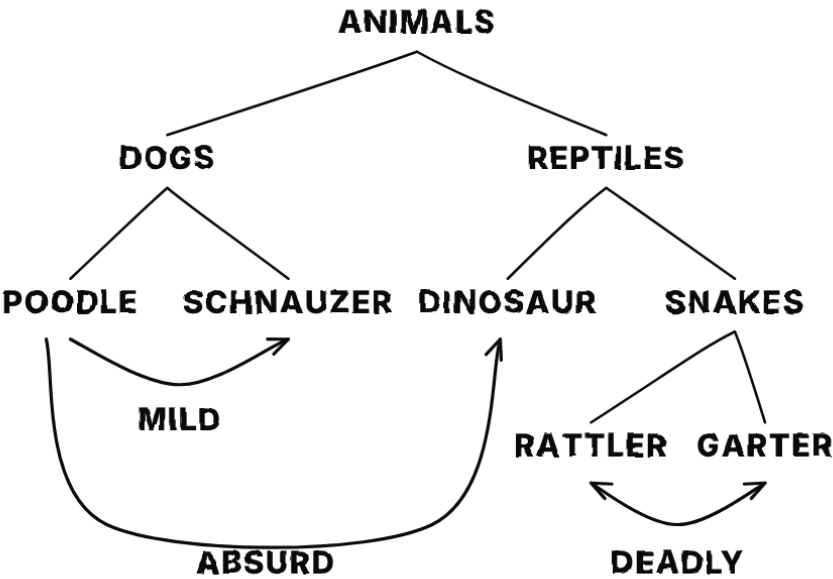
Likely is not the same as safe

Try it yourself

Next time an app or a website gives you a confidence figure, ask yourself what it would cost if the less likely answer were true. You will find that a ninety percent "yes" can be plenty for choosing a song, and nowhere near enough for crossing a road or taking a medicine.

Classification can be more subtle still, because not all mistakes are equal. Sometimes classes form a family tree, the way the naturalist Linnaeus arranged animals into nested groups, and then mixing up close relatives is a smaller error than mixing up distant ones.

Mistaking a poodle for a schnauzer is forgivable; mistaking a poodle for a dinosaur is absurd. This is called hierarchical classification. But which tree matters depends on what the model is for. A rattlesnake and a harmless garter snake are close relatives, yet confusing the two could be deadly.



Some mistakes are worse

Remember

Classification answers "which one?": it puts each example into one of a fixed set of classes. Two classes make binary classification, more than two make multiclass classification. Models usually give a probability for each class, which also shows how sure they are. The most likely class is not always the one to act on: a twenty percent chance of a death cap is reason enough not to eat the mushroom. When classes form a family tree, some mistakes are worse than others, and which matter most depends on the use, as with rattlesnakes and garter snakes.

Questions

1. Which of these is a classification problem rather than regression?
 - a. How much will this house sell for?
 - b. Which digit is written in this patch of a cheque?
 - c. How many hours will this repair take?
 - d. How much rain will fall tomorrow?
2. An app is eighty percent sure a mushroom is not a death cap. Why should you still not eat it?
 - a. Because eighty percent is the least likely answer
 - b. Because the app was trained on too few mushrooms
 - c. Because a twenty percent chance of a deadly outcome outweighs one good dinner

- d. Because mushrooms cannot be classified from photos
3. Why is mistaking a rattlesnake for a garter snake worse than their closeness in the family tree suggests?
- a. Because the consequences of the mistake, not just the closeness of the classes, decide how bad it is
 - b. Because snakes cannot be classified hierarchically
 - c. Because the model gave both the same probability
 - d. Because rattlesnakes are rarer in the data
4. Why do classifiers usually give a probability for each class instead of one firm answer?
- a. Because firm answers take longer to print
 - b. Because probabilities are always correct
 - c. Because users prefer numbers
 - d. Because a model that only gives a flat answer is hard to train, and probabilities also show how sure it is

Answers

1. **b)** Classification answers "which one?": deciding which character out of a known set a patch shows.
2. **c)** To decide, you weigh each choice by how likely each outcome is and how good or terrible it would be; a one-in-five chance of dying is not worth a dinner.
3. **a)** Which tree matters depends on what the model is for; the two snakes are close relatives, yet confusing them could be deadly.
4. **d)** A model that can only say a flat cat or dog is hard to train, so it gives a probability for each class, which also gives a sense of how unsure it is.

Practice

1. Sort these into binary, multiclass or regression: spam or not spam; which of the ten digits is on a cheque patch; how long a patient stays in hospital; which letter or digit a handwritten character is.

Hint: Is the answer a number, one of two classes, or one of many?

Answer: Spam or not spam is binary classification. The ten digits, and letters plus digits, are multiclass classification. Length of hospital stay is a number, so it is regression.

2. A weather app says there is a ninety percent chance of no rain. Explain why you might happily leave your umbrella for a walk, but not leave a valuable painting out in the garden overnight on the same forecast.

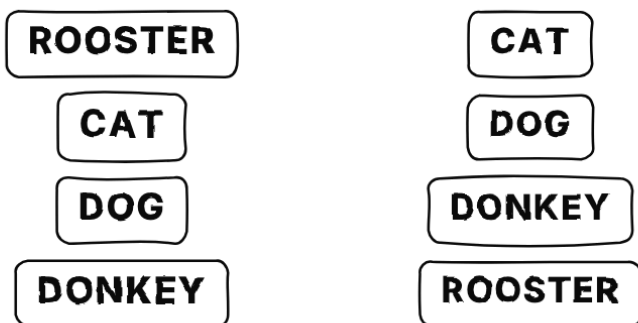
Hint: Weigh how likely the bad outcome is against how bad it would be in each case.

Answer: The chance of rain is the same, one in ten, but the harm is not. Getting a little wet on a walk is a small cost, so the risk is fine; a ruined painting is a large loss, so even a one-in-ten chance is not worth taking. As with the mushroom, the decision depends on both the probability and the consequences.

Tags, rankings and your feed

Picture the old German fairy tale of the Town Musicians of Bremen: a donkey, with a dog standing on its back, a cat on the dog, and a rooster on top. Show that picture to a program trained to answer "cat or dog?" and it is stuck. It must pick one, yet both are there, and a donkey and a rooster besides. However good the program is, the question it was built to answer is the wrong one for this picture.

MULTI-LABEL



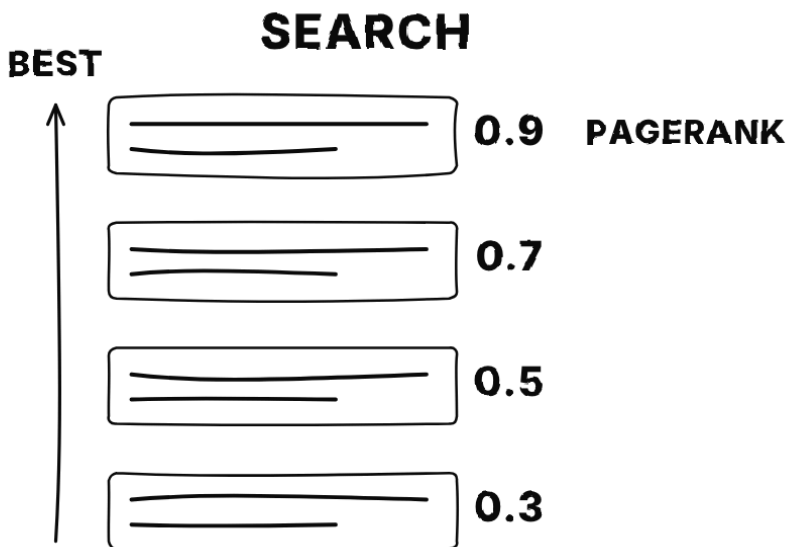
One picture, several labels

When an example can belong to several classes at once, the job is called multi-label classification. Instead of choosing one answer, the

model may say: a cat, and a dog, and a donkey, and a rooster. This is how automatic tagging works. A post on a technology blog might carry five to ten tags, such as "machine learning", "programming languages", "cloud computing". Tags tend to come in company: posts about cloud computing often mention Amazon's cloud service, and posts about machine learning often mention the graphics chips used to train it.

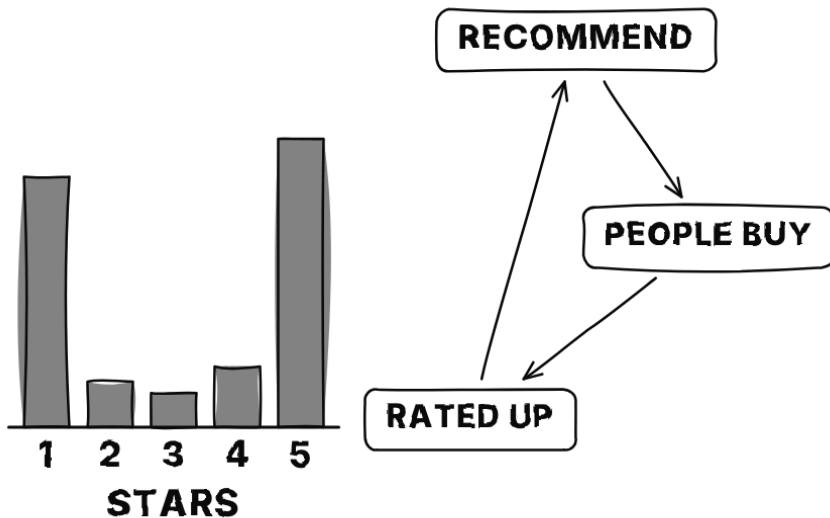
Sometimes the set of possible tags is enormous. At America's National Library of Medicine, trained staff tag every article added to its great medical index with terms from a list of about twenty-eight thousand subject headings. The tags let researchers find everything written on a question, but the work is slow, and an article may wait about a year for them. Machine learning can supply provisional tags in the meantime, until a person checks them, and there have been competitions for years on doing this well.

Search is a related problem, but the question changes. For a web search, there are usually many pages that are relevant; what matters is which of them to show at the top. One way is to give every page a score and show the highest scores first. Google's original secret ingredient, PageRank, was such a score, and oddly it did not depend on what you had typed at all. A simple filter picked out the pages that matched your words, and PageRank then put the more authoritative pages first. Today search engines use machine learning and models of how people behave to score each page for the particular search.



Search ranks by score

Recommendation is a close cousin of search, with one big difference: it is personal. A science fiction fan and a lover of old comedies should see quite different film suggestions, and the same goes for shopping, music and news. Sometimes people tell the system what they think, with star ratings and reviews. More often they tell it only by what they do, such as skipping a song, which may mean they dislike it, or just that it did not suit the moment. When your recommendation feed reshuffles after you skip a few videos, it is learning from exactly this kind of quiet signal. In the simplest form, the system learns a score for each item, such as the expected star rating or the chance you will buy it, and shows you the items with the highest scores. The book even shows a page of deep learning books that a large online shop suggested to one of its authors.



Lopsided ratings, self-feeding loops

Built naively, such systems have deep flaws. First, the feedback is lopsided: people mostly rate what they loved or hated, so ratings pile up at one star and five stars, with strangely few in the middle. Second, what people buy today is partly the result of what the system recommended yesterday, and the learning method may not allow for that. So a loop can form: the system pushes an item, more people buy it, the purchases make it look better, and the system pushes it even harder. This is why a feed can seem to show you more and more of the same thing once you have clicked on it a few times. How to deal with lopsided feedback, with people's incentives and with these loops are still open research questions.

Remember

Multi-label classification lets an example belong to several classes at once, like the four animals in one picture, and it is how automatic tagging works. Tag sets can be huge, like the twenty-eight thousand medical headings, where machine learning offers provisional tags. Search ranks relevant results by a score and shows the best first; PageRank was an early score that ignored what you typed. Recommendation is personal ranking, learned from ratings and from what people do. Lopsided ratings and loops where recommendations feed their own success make recommendation harder than it looks.

Questions

1. Why does a cat-or-dog program fail on the picture of the Town Musicians of Bremen?
 - a. Because the picture is a drawing, not a photo
 - b. Because it must choose one class, while several animals are in the picture at once
 - c. Because donkeys are harder to recognise than cats
 - d. Because it was not accurate enough
2. How did PageRank, Google's early score, differ from today's search scores?
 - a. It used star ratings from users
 - b. It scored each page for the particular words typed

- c. It did not depend on what was typed; a separate filter picked the matching pages
 - d. It ranked pages at random
3. Why do ratings often pile up at one star and five stars?
- a. Because people mostly rate things they loved or hated
 - b. Because the recommender hides three-star ratings
 - c. Because most products are either perfect or broken
 - d. Because star scales only have two values
4. What is the feedback loop the book warns about in recommendation?
- a. Users rating the same item twice
 - b. Ratings being lost when a song is skipped
 - c. The system recommending only items nobody has bought
 - d. The system pushes an item, more people buy it, so it looks better and is pushed even more

Answers

1. **b)** It must pick one, yet both a cat and a dog are there, and more; the question it was built to answer is the wrong one for this picture.
2. **c)** PageRank oddly did not depend on what you had typed; a simple filter picked the matching pages and PageRank put the more authoritative ones first.
3. **a)** The feedback is lopsided: people mostly rate what they loved or hated, so middle ratings are strangely few.
4. **d)** The system pushes an item, more people buy it, the purchases make it look better, and the system pushes it even harder.

Practice

1. For each case, say whether one class or several can be true at once: a photo of a single pet, cat or dog; a blog post about cloud computing and programming; a medical article that covers diabetes and the heart.

Hint: Can the example belong to more than one class at the same time?

Answer: The single pet is ordinary classification with one class. The blog post and the medical article can each carry several tags at once, so they are multi-label classification, which is how automatic tagging works.

2. You skip three songs in a row on a music app. Explain why the app cannot be sure you dislike them, and what kind of feedback this is.

Hint: What else might skipping mean?

Answer: Skipping is feedback given by what you do rather than by a rating. It may mean you dislike the songs, but it may only mean they did not suit the moment, so the app has to treat the signal with care.

When order matters

Pause a film at a random moment and look at the single frame. A man's arm is raised; is he waving, throwing, or about to strike? Play a second before and a second after, and the answer is obvious. One frame alone can mislead; the frames around it tell the story.

FRAMES AROUND IT



ANY LENGTH



PATIENT HISTORY

One frame misleads, frames tell

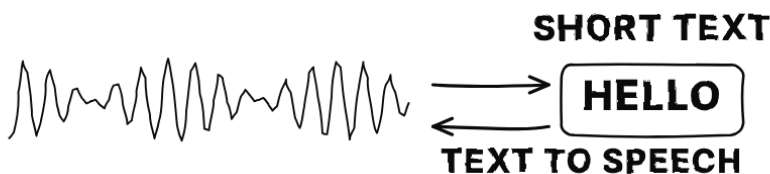
Everything so far has had a fixed shape. A house came as the same short list of facts, a photo as the same grid of pixels, and the answer was one price, or one probability per class. And each example was forgotten the moment the model had answered it, as if every case had

nothing to do with the one before. For houses that is fine, but many problems do not work that way.

A video clip can have any number of frames, and what is happening in one frame is far clearer when the model also looks at its neighbours. Language is the same: in machine translation, a whole sentence goes in, in one language, and a whole sentence comes out, in another. In a hospital's intensive care unit, a model might watch patients and raise an alarm whenever the risk of death in the next day passes some limit. It would be foolish to forget each patient's history every hour and judge them only on the latest readings. Problems like these are called sequence learning, because the model takes in sequences, or gives them out, or both. When both the input and the output are sequences of any length, it is called sequence-to-sequence learning.

TOM HAS DINNER IN
ENT — — —

WASHINGTON WITH SALLY
ENT — ENT



THOUSANDS OF NUMBERS

Tags in order; long sound, short text

The book picks out a few special cases. The first is tagging words in a sentence, where every word gets one label, in the same order. One might mark each word as noun, verb and so on, or mark which words name people, places or organisations: in "Tom has dinner in Washington with Sally", the names Tom, Washington and Sally get tagged.

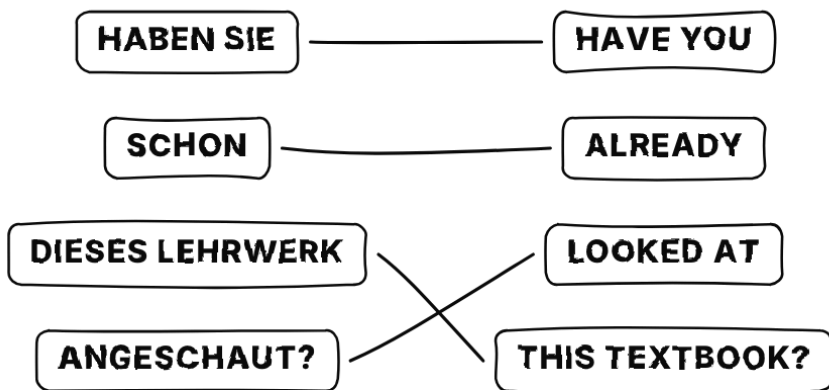
The second is speech recognition: a recording goes in and a written transcript comes out. The difficulty is that sound is measured thousands of times a second, so a single spoken word can stretch over thousands of measurements, and there is no neat one-to-one match between sound and letters. The output is far shorter than the input. When your voice assistant writes out your question on the screen, it

has just solved this problem, squeezing a long stream of sound into a handful of words. People recognise speech well even from a crackly phone line, but for computers it is a formidable challenge. The reverse task, text to speech, turns a short text into a long sound recording.

The third is machine translation, which adds a new twist: the pieces need not come in the same order. German, for instance, likes to put the verb at the end. "Haben Sie sich schon dieses grossartige Lehrwerk angeschaut?" means "Have you already looked at this excellent textbook?" Translated word by word in German order, it comes out as "Have you yourself already this excellent textbook looked at?", which no English speaker would say. The chatbot you type to faces all of this at once, and more: to decide what to say next, it has to keep track of the whole conversation so far and draw on knowledge of the world. Working out the order in which someone reads a web page, or holding a long conversation, are still active areas of research.

GERMAN

ENGLISH



VERB AT THE END

Translation changes the order

Remember

Many problems do not come in fixed shapes: videos, sentences and patient records unfold over time, and each part makes sense only alongside the others. Sequence learning handles inputs or outputs that are sequences, and sequence-to-sequence learning handles both, of any length. Tagging gives each word one label in order; speech recognition squeezes long sound into short text, and text to speech does the reverse. Translation is harder still, because the matching pieces may come in a different order, like German verbs at the end of a sentence.

Questions

1. Why would it be foolish for an intensive care model to judge each patient only on the latest readings?
 - a. Because the latest readings are always wrong
 - b. Because the patient's history over time helps show the real risk
 - c. Because the model can only read one number at a time
 - d. Because alarms must be raised every hour
2. What makes speech recognition hard compared with tagging words in a sentence?
 - a. A word can stretch over thousands of sound measurements, so sound and letters do not match one to one
 - b. Speech always comes in a fixed length
 - c. Every word must be given a part-of-speech label
 - d. The output is longer than the input
3. What new difficulty does machine translation add?
 - a. The input has no words in it
 - b. The output is always shorter than the input
 - c. Matching pieces may come in a different order, like German verbs at the end
 - d. Each word gets exactly one label

Answers

1. **b)** It would be foolish to forget each patient's history every hour and judge them only on the latest readings.
2. **a)** Sound is measured thousands of times a second, so a single spoken word stretches over thousands of measurements and there is no neat one-to-one match.
3. **c)** In translation the pieces need not come in the same order; German, for instance, likes to put the verb at the end.

Practice

1. For each task, say whether the output is about as long as the input, much shorter, or much longer: tagging each word of a sentence; turning a recording into text; reading a text aloud.

Hint: Compare how many pieces go in with how many come out.

Answer: Tagging gives one label per word, so the output is as long as the input. Speech recognition turns thousands of sound measurements into a few words, so the output is much shorter. Text to speech turns a short text into a long recording, so the output is much longer.

2. Explain with the paused film why a model that forgets each example after answering is a poor fit for video.

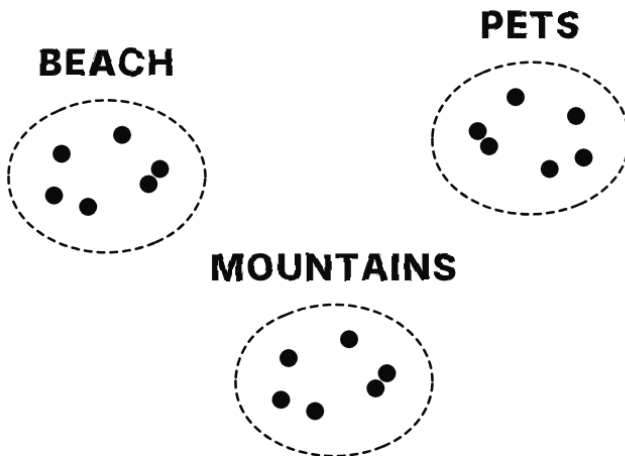
Hint: What helps you tell a wave from a throw?

Answer: One frame alone can be ambiguous, like a raised arm that might be waving or throwing. The frames before and after make it clear, so a model that judges each frame on its own, forgetting the others, throws away exactly the context it needs.

Learning without answers

Scroll through the photo gallery on your phone and look for the albums it made by itself: beaches, pets, mountains, the same few faces. Nobody told it which photo was a beach. It found the groups on its own, from nothing but the pictures. How can a machine learn anything when no one gives it the answers?

CLUSTERING



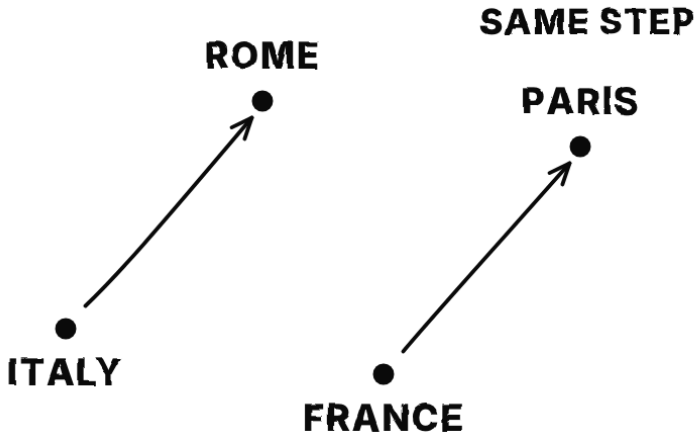
Groups found without answers

Supervised learning, the book says, is like a job with a very bossy boss who stands at your shoulder and tells you exactly what to do in every situation. Dull, perhaps, but easy to please: spot the pattern and copy

the boss. The opposite job is a boss who hands you a mountain of data and says, "Do some data science with it!" That is vague because it is vague. Learning from data without any answers attached is called unsupervised learning, and the questions you can ask are limited only by your imagination.

The book lists a few. Can the data be sorted into a handful of natural groups, such as photos of landscapes, dogs, babies, cats and mountain peaks, or shoppers who browse in similar ways? That is called clustering. Can a few numbers capture what matters? A thrown ball's path is well described by its speed, its size and its weight, and tailors fit clothes using a small set of body measurements. Can words be placed like points on a map, so that relations between them become directions, so that going from Italy to Rome is the same step as going from France to Paris? Can we find the hidden causes behind what we see, such as how house prices, pollution, crime, schooling and wages affect one another?

A MAP OF WORDS

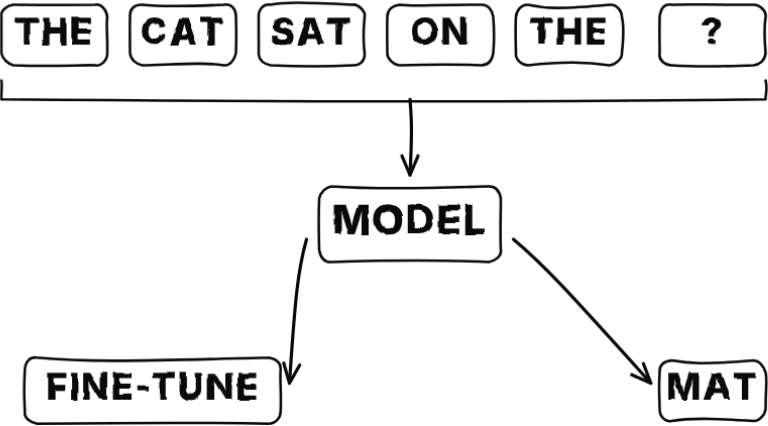


Words as points on a map

One of the most exciting recent developments is generative models. They learn what the data looks like so well that they can judge how likely an example is, or produce brand new examples of their own. The image generators that paint a picture from your description belong to this family; the book names diffusion models among the latest advances.

Another development is self-supervised learning, where the data itself provides the questions and the answers. For text, hide a word at random and ask the model to guess it from the words around it; huge amounts of text can be used this way with no labelling at all. The chatbot you type to started from training of this kind, learning to predict missing or next words across an enormous amount of text. For

pictures, a model can be asked where one cut-out piece sat relative to another, to fill in a covered-up part, or to tell whether two altered images came from the same original. What a model learns this way can then be reused: it is fine-tuned, with a little extra training, for the particular task you actually care about.



Hide a word, guess it back

Remember

Supervised learning has a bossy boss with all the answers; unsupervised learning gets only the data and must find structure in it. It can find natural groups, a few numbers that capture what matters, relations between words, and hidden causes. Generative models learn the data so well that they can produce new examples of their own. Self-supervised learning makes its own questions from the data, such as guessing hidden words, and what it learns is then fine-tuned for the task we need.

Questions

1. What is the key difference between supervised and unsupervised learning?
 - a. Unsupervised learning uses less data
 - b. In unsupervised learning the data comes with no answers attached, and the model must find structure itself
 - c. Unsupervised learning needs a human watching it
 - d. Supervised learning cannot handle photos
2. Your phone sorts photos into beaches, pets and mountains without being told which is which. What is this called?
 - a. Clustering
 - b. Fine-tuning
 - c. Regression
 - d. Classification with labels

3. How does self-supervised learning get its questions and answers without anyone labelling data?
 - a. It copies the answers of a supervised model
 - b. It asks users to rate examples
 - c. It hides part of the data, such as a word, and learns to guess it from the rest
 - d. It only works on data that was labelled long ago
4. What happens after a model has learned from self-supervised training?
 - a. It is thrown away
 - b. It is used only to guess hidden words
 - c. It becomes a clustering model
 - d. It is fine-tuned with a little extra training for the task we care about

Answers

1. **b)** Learning from data without any answers attached is called unsupervised learning.
2. **a)** Sorting data into a handful of natural groups, such as landscapes, dogs and cats, is called clustering.
3. **c)** For text, hide a word at random and ask the model to guess it from the words around it, with no labelling at all.
4. **d)** What a model learns this way can be reused: it is fine-tuned with a little extra training for the particular task you care about.

Practice

1. Say whether each is clustering, a generative model, or self-supervised learning: grouping shoppers by how they browse; an app that paints a picture from your description; guessing a hidden word from the words around it.

Hint: Is the task to find groups, to make new examples, or to answer a question made from the data itself?

Answer: Grouping shoppers is clustering; painting a picture from a description is the work of a generative model; guessing a hidden word is self-supervised learning.

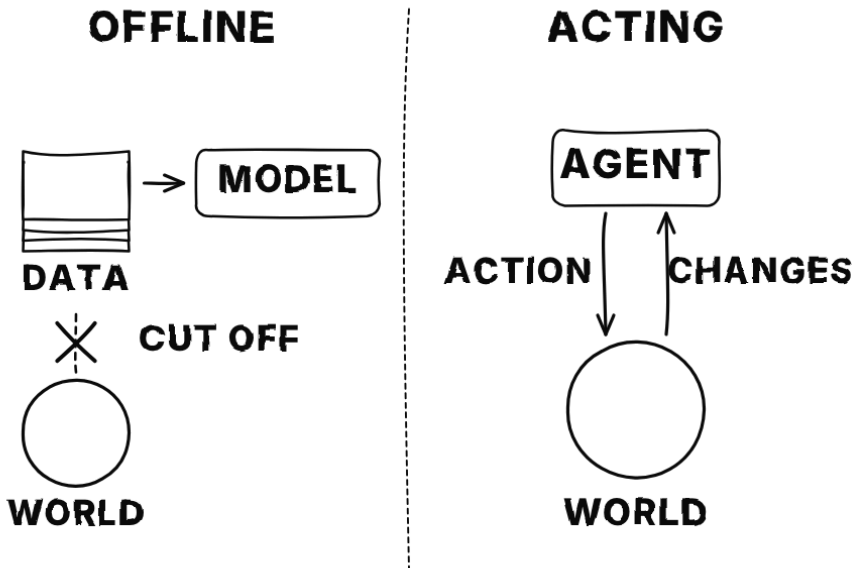
2. Explain in your own words why tailors' measurements are a good example of finding "a few numbers that capture what matters".

Hint: How many numbers does it take to describe a body well enough to fit clothes?

Answer: A human body is very complicated, yet a small set of measurements, such as chest, waist and length, describes its shape well enough to fit clothes. In the same way, unsupervised learning can look for a few numbers that capture the important properties of complicated data.

Learning by doing

Watch a small child learn to stack blocks. Nobody hands them a list of right answers. They put one block on another, the tower wobbles or falls or stands, and they try again. Each move changes what they see next, and the only verdict is whether the tower stays up in the end.

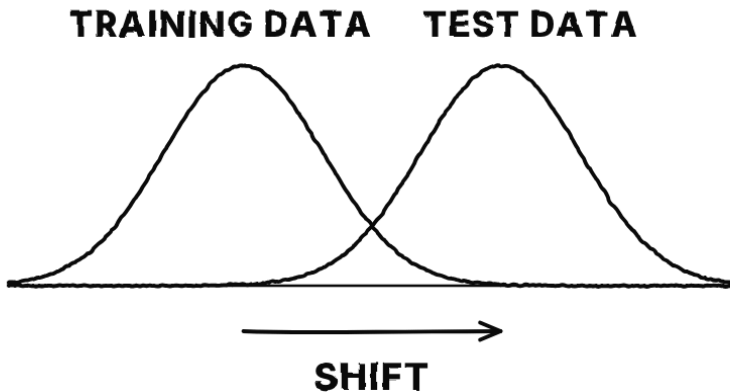


Learning cut off, or acting

Everything so far has worked differently. We grabbed a big pile of data up front, switched off contact with the world, and let the pattern-finding machine get on with it. Because all the learning happens after the machine is cut off from the world, this is called offline learning. It

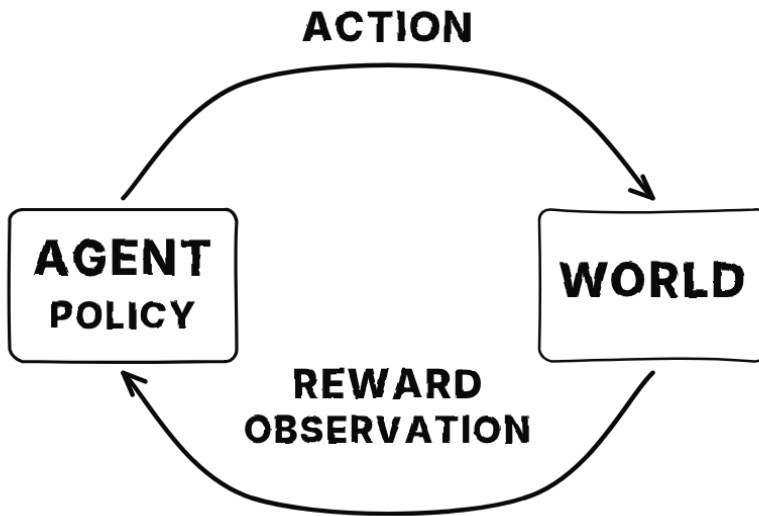
has its charms: we can think about recognising patterns without worrying about a world that moves and responds. But it is limiting. Readers of Asimov's robot stories picture machines that do not just predict but act, and actions, unlike predictions, change the world the machine will see next. To train such an agent, we must take into account how its own actions shape what comes next, and offline learning cannot.

Once the world talks back, new questions appear. Does it remember what we did before? Does it want to help us, like someone speaking clearly into a speech recogniser? Does it want to beat us? Spammers are the book's example: they keep changing their emails to slip past your spam filter. Does it change over time, on its own or because of our tools? Whenever the data at test time differs from the data at training time, we have what is called distribution shift. The book's example will be familiar from school: doing homework set by the teaching assistants, and then sitting an exam written by the lecturer.



Test data can drift

The main framework for machines that act is reinforcement learning. It is used in robots, in conversational systems and in the computer opponents of video games. Combined with deep learning it has had famous successes: a program that learned to beat people at old Atari arcade games from the screen alone, and AlphaGo, which defeated the world champion at the board game Go. The picture is a loop. At each step, the agent observes the world, chooses an action, and the world responds, sometimes with a reward; then the agent observes again, and so on. The agent's way of choosing actions from what it sees is called its policy, and the aim of reinforcement learning is to find a good policy.

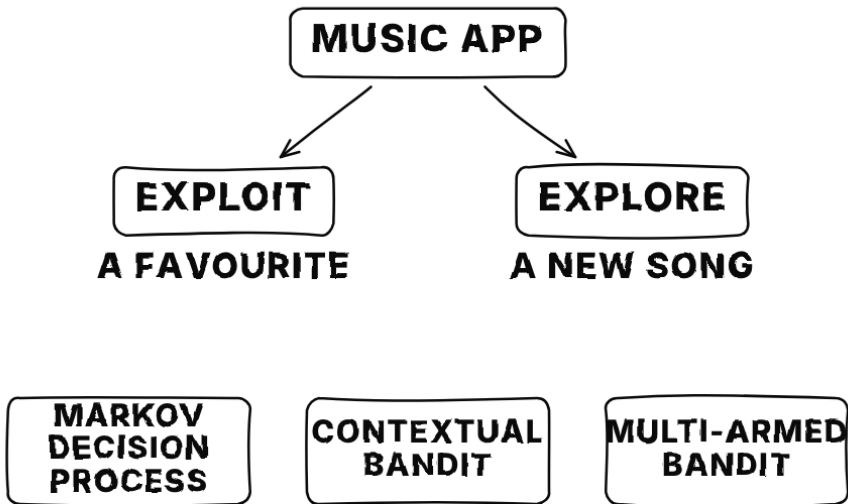


The reinforcement learning loop

The framework is extremely general; even ordinary classification can be dressed up as reinforcement learning, with one action for each class and the reward set by how right the choice was. But it can do far more than supervised learning, because the world rarely tells the agent the best action. It gives only a reward, and often not even which action earned it. In chess, the only real verdict comes at the end: win or lose. Which of the forty moves deserves the credit or the blame? This is the credit assignment problem, and people face it too: someone promoted at work has to work out which of the past year's choices actually led to it.

There are further difficulties. The agent may see only part of the situation, like a cleaning robot stuck in one of several identical

cupboards, which has to remember how it got there to know where it is. And it must choose between using the best strategy it knows and trying something new that might be better, giving up some reward now for knowledge. That is the trade-off between exploiting and exploring. Your music app faces it every day: play songs it knows you like, or risk a new one you may love or skip.



Exploit or explore; simpler cases

Tackling all of this at once may be asking too much, and not every real problem has all of it, so researchers study simpler special cases. When the agent can see the whole situation, the problem is called a Markov decision process. When the situation does not depend on what the agent did before, it is a contextual bandit problem. And when there is no situation at all, only a row of possible actions with unknown

rewards, like a row of slot machines, it is the classic multi-armed bandit problem.

Remember

Offline learning collects data first and learns afterwards, cut off from the world; an agent that acts changes the world it will see next. When test data differs from training data, as when the exam is written by someone other than the homework setter, that is distribution shift. Reinforcement learning is a loop of observing, acting and getting rewards, and its goal is a good policy for choosing actions. Its hardest problems are knowing which actions earned a late reward, seeing only part of the situation, and balancing exploring with exploiting. Simpler special cases are the Markov decision process, the contextual bandit and the multi-armed bandit.

Questions

1. Why is offline learning not enough for training an agent that acts?
 - a. Because offline learning cannot use large datasets
 - b. Because an agent's actions change what it will see next, which a fixed pile of data cannot capture
 - c. Because agents never receive rewards
 - d. Because offline learning only works on text

2. Homework is set by teaching assistants, but the exam is written by the lecturer. What machine learning problem does this illustrate?
 - a. Credit assignment
 - b. Exploration
 - c. Distribution shift: test data differing from training data
 - d. A multi-armed bandit
3. A chess program wins a game. What is hard about learning from this one reward?
 - a. Working out which of its many moves deserve the credit
 - b. Knowing whether it won
 - c. Seeing the board
 - d. Choosing an opponent
4. Your music app sometimes plays a song you have never heard instead of a known favourite. Why might it do this?
 - a. Because it forgot your favourites
 - b. Because it has no policy
 - c. Because the song was more popular
 - d. Because it trades some sure reward now for knowledge that might find something better

Answers

1. **b)** To train an agent, we must take into account how its own actions shape what comes next, and offline learning cannot.
2. **c)** Whenever the data at test time differs from the data at training time, we have distribution shift; the homework and the exam are the book's

example.

3. a) The only real verdict comes at the end, so the program must work out which moves deserve credit or blame: the credit assignment problem.

4. d) It must choose between using the best strategy it knows and trying something new: the trade-off between exploiting and exploring.

Practice

1. Describe the reinforcement learning loop for a robot vacuum cleaner: what are its observations, its actions and its rewards, and what is its policy?

Hint: Observe, act, get a reward, observe again.

Answer: The robot observes its surroundings through its sensors, chooses actions such as moving forward or turning, and receives rewards, for example for cleaning dirt, or penalties for bumping into things. Its policy is its rule for picking an action from what it currently observes, and learning aims to make that policy good.

2. A cleaning robot is stuck in one of several identical cupboards. Explain why it cannot tell where it is from what it sees right now, and what it could use instead.

Hint: What does the inside of each cupboard look like?

Answer: All the cupboards look the same from inside, so the current view does not tell it which one it is in; it sees only part of the situation. It has to use what it saw earlier, before it went in, to work out its location.

Old roots

Picture a Sunday morning five hundred years ago, outside a German church. As the congregation comes out, sixteen grown men are asked to stand in a line, heel to toe. The total length of their feet is measured and shared out equally among the sixteen, and the result is declared to be one foot. It sounds quaint, yet it is a real method of learning from data, and it appears in a sixteenth-century geometry book by Jacob Köbel.

A TYPICAL FOOT

SHORTEST

LONGEST



$$\text{total} \div 16$$

$$\text{rest} \div 14$$

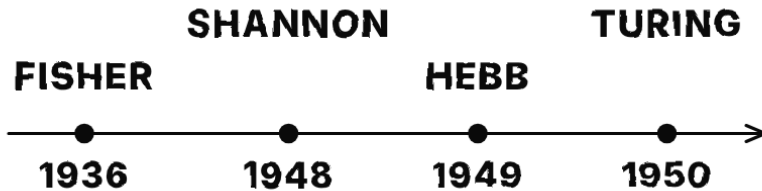
TRIMMED MEAN

Sixteen feet, a trimmed mean

Deep learning is young, but the wish to learn from data and predict what comes next is very old, and it lies at the root of much of science and mathematics. Two great mathematicians lent their names to patterns of chance still used everywhere: Jacob Bernoulli, around the year 1700, and Carl Friedrich Gauss, a century later. Gauss also invented the method of least squares, which fits a line to measurements by keeping the misses small, and which is still used for everything from insurance to medical diagnosis. Such tools sharpened experimental science; Ohm's law, which links the current through a wire to the voltage across it, is a perfect straight-line rule.

The foot-measurers even improved their method. To guard against unusual feet, the men with the shortest and the longest feet were sent away, and only the rest were averaged. This is one of the earliest known examples of a trimmed mean, an average that ignores the extremes. Some rating systems use the same trick, ignoring the very highest and lowest scores so that a few wild opinions do not swing the result.

Statistics truly took off once data was being collected on a large scale. One of its pioneers, Ronald Fisher, built much of its theory and applied it to genetics, and several of his methods are still standard. A small dataset he published in 1936, measurements of iris flowers, is still sometimes used to show off learning methods today. Fisher was also a supporter of eugenics, a reminder that the misuse of data science has as long a history as its good uses.

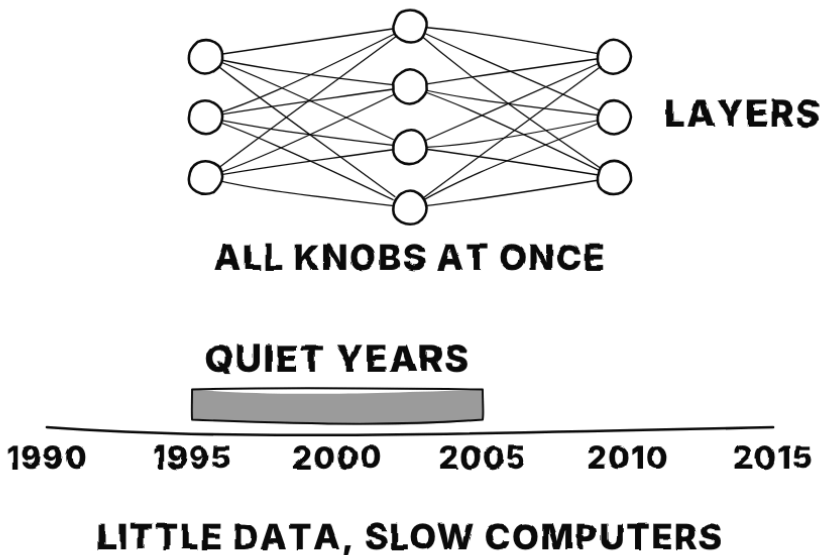


Old roots of learning from data

Other roots came from elsewhere. Claude Shannon founded the theory of information, and Alan Turing the theory of computation. Turing asked, "Can machines think?", and proposed what is now called the Turing test: a machine counts as intelligent if a person exchanging written messages with it cannot easily tell it from a human. Today's chatbots have made that old question feel very real.

The study of the brain added more. The psychologist Donald Hebb suggested that connections between nerve cells grow stronger when they are used successfully, a rule now named after him. That idea led to later learning methods and to the basic recipe of deep learning today: strengthen what works, weaken what does not. It is also why these systems are called neural networks, after networks of nerve

cells, though the resemblance to biology has grown looser over time. At their heart are two ideas: layers of simple processing stacked one after another, and a way of adjusting all the knobs of the whole network at once, called backpropagation.



Layers, and the quiet years

After a burst of early progress, research on neural networks stalled from about 1995 to 2005. Training them took a great deal of computing power, which was scarce, and datasets were small: a collection of sixty thousand handwritten digits was thought huge, and Fisher's 1936 flowers were still in use. With so little data and computing, other methods, with names like kernel methods and decision trees, worked better in practice. They also trained in hours rather than weeks and came with solid mathematical guarantees.

Remember

Learning from data is centuries old, from averaging sixteen men's feet to Gauss's least squares. Throwing out the extremes before averaging, the trimmed mean, is one of its earliest refinements. Statistics, information theory, the theory of computation and the study of the brain all fed into machine learning, and its misuse is as old as its good use. Neural networks rest on stacked layers and on adjusting all their knobs at once, but they stalled for a decade because computing was scarce and datasets small.

Questions

1. Why did the foot-measurers send away the men with the shortest and longest feet?
 - a. To make the line shorter
 - b. To keep unusual feet from distorting the average
 - c. Because only fourteen men were needed
 - d. To make the foot longer
2. What does the story of Ronald Fisher remind us of?
 - a. That statistics began only in the twentieth century
 - b. That small datasets are useless
 - c. That data science can be misused, and has been for as long as it has been used well
 - d. That flowers are the best data

3. Why did research on neural networks stall from about 1995 to 2005?
- a. Computing power was scarce and datasets were small
 - b. Neural networks had been proved not to work
 - c. Backpropagation had not yet been invented
 - d. Researchers lost interest in the brain

Answers

1. **b)** To guard against unusual feet, the extremes were sent away and only the rest averaged: a trimmed mean.
2. **c)** Fisher supported eugenics, a reminder that the misuse of data science has as long a history as its good uses.
3. **a)** Training took a great deal of computing power, which was scarce, and datasets were small.

Practice

1. Five judges score a dive: 6, 7, 7, 8 and 1. Explain why a trimmed mean, dropping the highest and lowest score, gives a fairer result than the ordinary average, and say which scores it keeps.

Hint: Which score is out of line with the others?

Answer: The score of 1 is far out of line and would drag the ordinary average down. Dropping the highest and lowest leaves 7, 7 and 7, so the trimmed mean is 7, which reflects what most judges thought.

2. Explain in your own words what the Turing test asks.

Hint: Think of a conversation carried out only in writing.

Answer: A person exchanges written messages with something that might be a machine or a human. If the person cannot easily tell the

machine's replies from a human's, Turing proposed that the machine should count as intelligent.

The great acceleration

Think of the first computer game you ever played, and then of a game on a phone today. The jump in what a small machine can draw, and how fast, is enormous. Here is the surprise: the chips built to make games look good turned out to be exactly what neural networks had been waiting for. That is one of the threads in how deep learning went from stalled to everywhere.

GROWTH SINCE 1970

DATA



COMPUTING



MEMORY



What grew, and how much

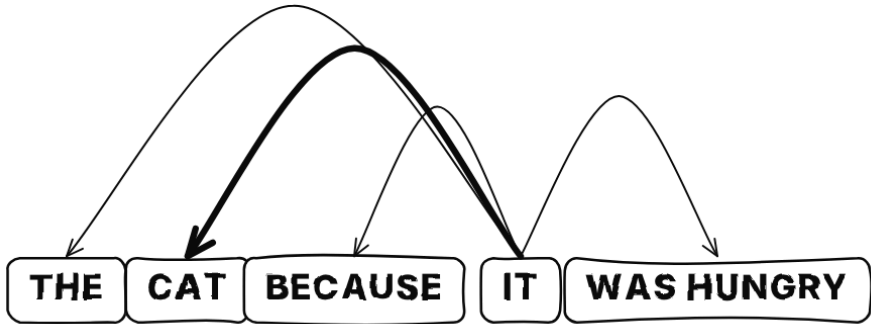
The change began with data. The web, companies serving hundreds of millions of people online, cheap and good sensors in everything, and

storage that kept getting cheaper together produced data in quantities no one had seen. At the same time computing kept getting cheaper, and the graphics chips designed for computer games, called GPUs, turned out to do the arithmetic of neural networks extraordinarily fast. Suddenly methods that had looked hopelessly slow were within reach.

The book shows this with a table, decade by decade. In 1970 a typical dataset was the hundred-odd iris flowers, and a computer had a tiny memory and did about a hundred thousand calculations a second. By 2020 a dataset might be a whole social network, trillions of items, and a single machine full of graphics chips could do roughly a million billion calculations a second. Memory grew too, but much less than the data, while computing power grew even faster than the data. So the methods that win are those that are thrifty with memory but happy to spend lots of computing on tuning their knobs, and that favoured deep neural networks over the older methods. It is also why several old ideas, from simple stacked networks to networks that remember sequences, were rediscovered after years on the shelf.

But the progress was not just old recipes on faster machines; the book compares the last decade to the Cambrian explosion, the burst in which animal life suddenly diversified. New ideas came thick and fast. One, called dropout, fights overfitting by deliberately adding random noise inside the network while it learns. Another, attention, lets a model look back at exactly the part of its input it needs, instead of cramming a whole sentence into memory before translating it, and long sentences became far more accurate.

ATTENTION



TRANSFORMER: ATTENTION ONLY

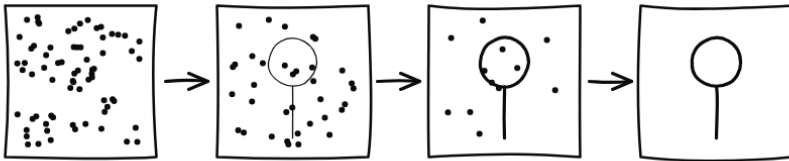
Attention looks back

Built entirely on attention, the Transformer design gets steadily better as the data, the model and the computing grow, and it now works for language, pictures, speech and much else. A single Transformer has even been trained to play Atari games, describe images, chat and control a robot. Language models, which predict text from other text, gained more and more abilities as they were scaled up. The chatbot you type to is one of these: the book names ChatGPT, a language model tuned to follow what people want, which can help fix computer code or write a story with you.

Generative models took leaps too. Generative adversarial networks pit two networks against each other: one makes fake examples, the other tries to tell fake from real, and the maker improves until the checker is

fooled, producing convincing faces of people who never existed. More recently, diffusion models learn to undo the gradual adding of noise, building a picture step by step out of static. Image generators that paint a picture from your description, such as the ones the book names, DALL-E 2 and Imagen, work this way.

DIFFUSION



REMOVE NOISE

Picture out of noise

Engineering mattered as much as ideas. When one graphics chip is not enough, training can be split across hundreds or thousands of them working together. That once-standard image-recognition task, which used to take days of training, was brought down to under seven minutes this way. Running many copies of a simulated world in parallel also powered reinforcement learning to beat the best humans

at Go, at Atari games and at the strategy game StarCraft, because such learners improve most when they can try things millions of times.

Finally, software made all this available to everyone. Free, shared toolkits for building neural networks spread ideas fast; the first generation had names like Caffe, Torch and Theano, followed by TensorFlow and others. The newest kind, such as PyTorch and JAX, let people write models in the same natural style as ordinary scientific programming in Python, which made experimenting far easier.

Remember

Deep learning revived when huge datasets met cheap storage and fast computing, above all graphics chips first made for games. Computing grew faster than data, and data faster than memory, which favoured deep networks and brought old ideas back. New ideas such as dropout, attention, the Transformer, language models, adversarial networks and diffusion models drove a burst of progress. Training spread across many chips and many simulations, and free toolkits like PyTorch put the methods within everyone's reach.

Questions

1. Why did graphics chips made for games matter so much to deep learning?
 - a. They do the arithmetic of neural networks extraordinarily fast

- b. They store more data than ordinary memory
 - c. They were cheaper than any other computer
 - d. They came with free software
2. Computing power grew faster than data, and data faster than memory. Which kind of method did this favour?
- a. Methods that store every example in memory
 - b. Methods thrifty with memory but happy to spend lots of computing tuning their knobs, such as deep networks
 - c. Methods that need no training at all
 - d. Methods that use only small datasets
3. What problem does attention solve in translating a long sentence?
- a. It makes the model smaller
 - b. It removes the need for training data
 - c. It lets the model look back at the part it needs instead of cramming the whole sentence into memory first
 - d. It adds noise to prevent overfitting
4. How do the two networks in a generative adversarial network work together?
- a. Both make pictures and the better one is kept
 - b. One translates, the other checks the grammar
 - c. They split the training data between them
 - d. One makes fakes, the other tries to spot them, and the maker improves until the checker is fooled

Answers

1. **a)** The graphics chips designed for games turned out to do the arithmetic of neural networks extraordinarily fast.
2. **b)** The methods that win are thrifty with memory but happy to spend lots of computing on tuning their knobs, which favoured deep neural networks.
3. **c)** Attention lets a model look back at exactly the part of its input it needs, instead of cramming a whole sentence into memory before translating.
4. **d)** One network makes fake examples, the other tries to tell fake from real, and the maker improves until the checker is fooled.

Practice

1. Explain in your own words why the book compares the last decade of deep learning to the Cambrian explosion, and give two of the new ideas behind it.

Hint: Was the progress only faster machines running old methods?

Answer: The Cambrian explosion was a sudden burst of new forms of life; the last decade saw a sudden burst of new methods, not just old recipes on faster machines. Examples are dropout, which fights overfitting with noise, and attention and the Transformer built on it, which let models handle long inputs and keep improving as they grow.

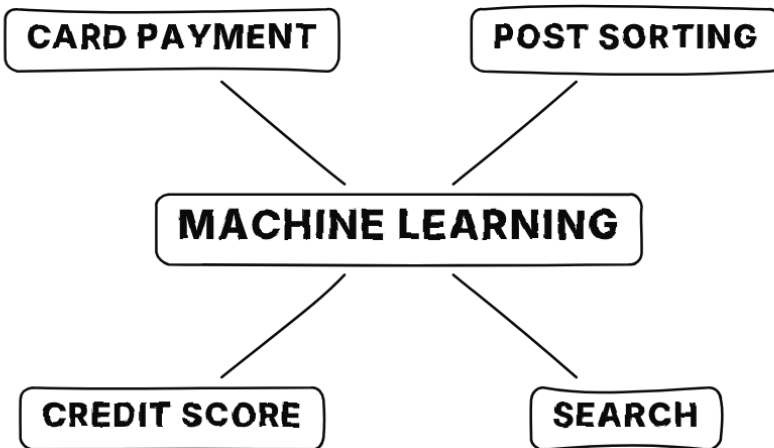
2. Why did running many simulated worlds in parallel help reinforcement learning beat people at Go and Atari games?

Hint: What does a reinforcement learner need plenty of?

Answer: Reinforcement learners improve by trying actions and seeing the rewards, and they need enormous numbers of tries. Simulations running in parallel supply millions of such tries far faster than the real world could.

Everywhere, mostly unseen

Pay for something online with a card, and in the blink before the payment goes through, a program has already judged whether it looks like fraud. Post a letter, and a machine has probably read the handwritten address. Neither feels like artificial intelligence, yet both are machine learning at work, and have been for years. Most of it, the book says, is simply hidden from sight.



Machine learning, out of sight

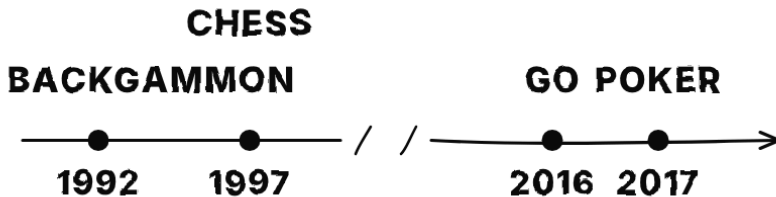
Part of the reason it spread so fast is that the tools became easy. Some people build better software, others build better models, and each side

benefits from the other's work. In 2014, training one fairly simple classifier was a homework problem worth setting to new doctoral students in machine learning at a top university. Today the same job takes fewer than ten lines of code, well within reach of any programmer.

Some successes are old. Machines have sorted post by reading handwritten addresses since the 1990s, which is where the famous collection of handwritten digits came from. They read cheques, score people's creditworthiness and check payments for fraud behind most of the big payment services. Chess programs have been strong for decades, and machine learning quietly drives search, recommendations and the personal ordering of what you see online.

What put artificial intelligence in the spotlight recently is that it began to solve problems people had thought impossible, problems ordinary consumers meet directly, and deep learning is behind many of them. Voice assistants such as Siri, Alexa and Google's assistant answer spoken requests reasonably well, from switching on a light to booking a haircut, and that is probably the most visible sign of artificial intelligence in daily life. Their key ingredient, speech recognition, has become as accurate as people for some uses. Recognising objects in photos has improved dramatically: on a standard test, the share of photos for which the right answer was not even among a model's top five guesses fell from more than a quarter in 2010 to barely more than two in a hundred by 2017. Similar leaps came in identifying birds by their songs and in diagnosing skin cancer.

GAMES WON BY MACHINES



Games machines learned to win

Games used to be a yardstick of human ability, and machines have climbed it step by step. A program learned backgammon by reinforcement learning; Deep Blue beat the world chess champion Garry Kasparov with special hardware and a fast search through possible moves. Go, with its vastly larger number of positions, fell to AlphaGo, which combined deep learning with a clever way of sampling possible futures. Poker was hard because you cannot see the other players' cards, and a program called Libratus still beat human professionals. Self-driving cars have made great progress too, with some cars already driving themselves part of the time, though full self-driving is still out of reach: driving needs seeing, reasoning and following rules, and deep learning mainly handles the seeing.

Robotics, logistics, biology, particle physics and astronomy all owe recent advances to machine learning as well.

HEADLINES

ROBOT UPRISING

NO SIGN OF IT

REAL CONCERNS

AUTOMATIC CREDIT

JOB'S AUTOMATED

BIASED MODELS

Headline fears, real concerns

Popular articles often worry about a coming robot apocalypse, machines becoming conscious and deciding against us. The authors are calm on this point. Today's systems are built and trained for specific goals, and behind any appearance of general intelligence lies a mixture of rules, rules of thumb and statistical models. There are, they say, simply no tools yet for a general intelligence that can understand itself and redesign itself to solve any task.

The real worries are closer to home. Artificial intelligence already shapes people's lives: credit is judged automatically, autopilots handle

much of the work of steering vehicles, and decisions about granting bail draw on statistical data. Many routine jobs done by people can and will be automated; farm robots may make organic food cheaper but also replace harvest workers, and because simple manual work employs huge numbers of people in many countries, the effects on society could be profound. And models used carelessly can be biased by race, gender or age, which is especially troubling when they help make decisions that matter. Like the hiring model in an earlier part, a system that decides who gets a loan can quietly copy the unfairness of the past. The authors judge these to be far more pressing concerns than an evil superintelligence, and they urge that such systems be used with care.

Remember

Machine learning has worked quietly for decades, sorting mail, catching fraud and ranking what we see online. Easy software tools now let any programmer do what was once a doctoral exercise. Deep learning brought visible leaps: voice assistants, speech and image recognition, skin cancer diagnosis, and victories at Go and poker. The authors see no sign of a conscious machine uprising; the real concerns are lost jobs and biased decisions, which call for careful use.

Questions

1. Why does the book say machine learning is often hidden from sight?

- a. Because companies keep it secret by law
 - b. Because it works quietly in everyday systems like fraud checks and mail sorting that do not feel like AI
 - c. Because it only runs at night
 - d. Because it has no practical uses yet
2. Why was poker a harder game for machines than chess in one particular way?
- a. Poker has more rules
 - b. Poker games last longer
 - c. Players cannot see each other's cards, so the situation is only partly visible
 - d. Poker needs special hardware
3. Which concern do the authors consider most pressing?
- a. Lost jobs and biased automated decisions
 - b. A conscious machine deciding against humanity
 - c. Computers running out of memory
 - d. Games becoming too easy

Answers

1. b) Fraud checks and address reading do not feel like artificial intelligence, yet both are machine learning at work: most of it is hidden from sight.

2. c) Poker was hard because you cannot see the other players' cards.

3. a) The authors judge job automation and bias far more pressing than an evil superintelligence.

Practice

1. List three ways machine learning affects your day without your noticing, using examples from the lecture.

Hint: Think of payments, post and what you see online.

Answer: A card payment is checked for fraud automatically; a handwritten address on a letter may be read by a machine; and search results, recommendations and the order of what you see online are set by machine learning.

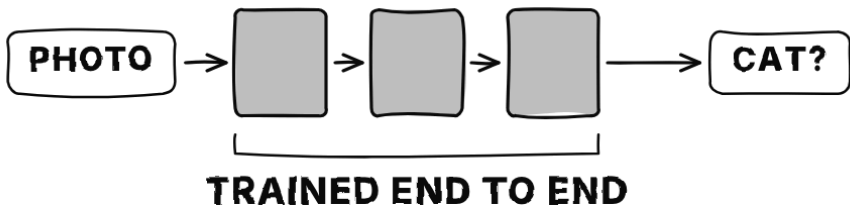
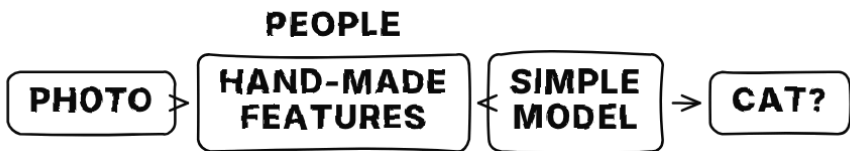
2. Explain in your own words why the authors are not worried about a conscious AI uprising, but are worried about bias.

Hint: What are today's systems actually built from, and what happens when they are used carelessly?

Answer: Today's systems are built and trained for specific goals from rules, rules of thumb and statistical models, and there are no tools for a machine that can understand and redesign itself. But carelessly used models can be biased by race, gender or age, and when they help decide about loans, bail or jobs, that unfairness harms real people now.

What makes it deep

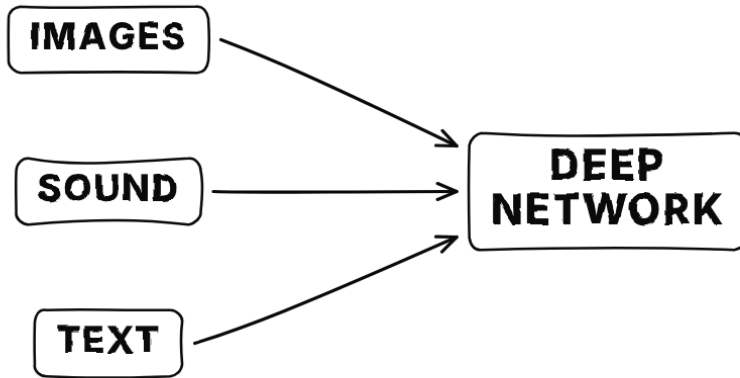
Imagine teaching a robot to recognise cats the old way. First you, a clever person, write rules to find edges, corners and patches of fur in a photo. Then you hand those findings to a simple learning machine that decides "cat" or "not cat". If your edge-finder misses something important, no amount of learning at the end can recover it. Deep learning throws that split away, and that is the heart of what makes it different.



Hand-made front, or end to end

Deep learning is the part of machine learning built on neural networks with many layers. It is deep in exactly that sense: its models learn many layers of steps, each one reshaping the data a little and handing it on. In a way, all machine learning has several stages, the first usually being some preparation of the features. What sets deep learning apart is that every one of its many layers is learned, all together, from the data.

That is why it shines exactly where older methods stumbled: raw sound, the raw pixels of photos, and sentences of any length to be turned into another language. The book singles out one thing most deep learning methods share, called end-to-end training. Instead of building a system from parts that are each tuned separately, you build the whole thing and tune all of it together. In computer vision, for over a decade the best results came from hand-designed feature finders, famous edge detectors and the like, that turned images into lists of numbers for a simple model to use. The crucial skill was inventing clever ways to prepare the data by hand. But there is only so much human ingenuity can do against a method that tries millions of options automatically, and when deep learning took over, those hand-made feature finders were replaced by filters tuned by the machine itself, with better accuracy.



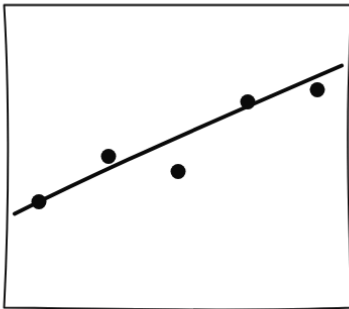
One kind of network for every field

So deep learning replaces not just the simple model at the end, but also the long, hand-crafted preparation before it. And because so much of that preparation was specific to one field, the walls between vision, speech, language, medical data and other areas have largely come down; the same tools now serve them all. That is why the same kind of model can sit behind both the voice assistant that hears you and the chatbot that writes back to you.

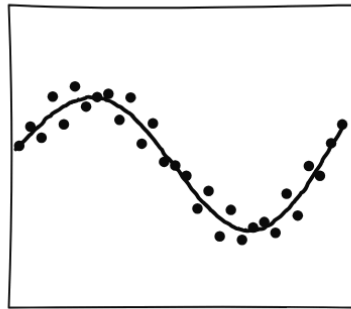
There are deeper shifts too. When data is scarce, models must lean on simplifying assumptions about the world; when data is plentiful, they can follow the data more freely. The book likens this to physics last century: once computers arrived, physicists could simulate how electrons behave directly instead of working out simplified

approximations by hand, gaining accuracy, though often losing some understanding of why. Deep learning also accepts solutions that are good rather than provably best, and is willing to try ideas before proving them, and this practical spirit, along with a flood of new talent, has driven rapid progress.

FEW: ASSUME



MANY: FOLLOW



Assume less when data is plenty

Finally, the community prides itself on sharing: tools, models and trained networks are released freely across universities and companies. In that spirit the book itself is free to read, share and use, and the authors hope it lowers the barrier for anyone wanting to learn. And that closes our tour of the introduction: from rules that never learn, through the many kinds of learning problem, to the data, chips and shared tools that made deep learning possible.

Remember

Machine learning is about systems that improve at a task with experience, usually data. Deep learning uses neural networks with many layers, all learned together from the data, end to end. It replaced both the simple models and the hand-crafted preparation of data, and brought vision, speech and language under one set of tools. Its progress came from abundant data, fast graphics chips and free, efficient frameworks, together with a practical willingness to try before proving.

Questions

1. What sets deep learning apart from other machine learning that also has several stages?
 - a. It uses more data than any other method
 - b. Every one of its many layers is learned, all together, from the data
 - c. It needs no computer chips
 - d. It only works on images
2. Why did machine-tuned filters beat hand-designed feature finders in computer vision?
 - a. Because human ingenuity cannot match a method that automatically tries millions of options
 - b. Because hand-designed finders were illegal
 - c. Because the filters needed no data

- d. Because edges do not matter in photos
3. Why have the walls between vision, speech and language largely come down?
- a. Because all three use the same data
 - b. Because computers became faster
 - c. Because deep learning replaced the field-specific hand-made preparation of data, so the same tools serve them all
 - d. Because researchers merged their universities

Answers

1. **b)** All machine learning has stages, but in deep learning every one of its many layers is learned together from the data.
2. **a)** There is only so much human ingenuity can do against a method that tries millions of options automatically.
3. **c)** Because so much of the preparation was specific to one field, replacing it let the same tools serve vision, speech, language and more.

Practice

1. Explain in your own words the difference between the old two-step way of recognising cats and end-to-end training.

Hint: Who designs the first step in each case?

Answer: In the old way, a person hand-designs rules to find edges and corners, and a simple learning machine only learns the last step; if the hand-made step misses something, learning cannot recover it. In end-to-end training, the whole system, including the first steps, is built and tuned together from data, so the machine finds its own useful features.

2. Name the three things the course credits for deep learning's recent progress, and say in a sentence what each contributed.

Hint: One is about examples, one about hardware, one about software.

Answer: Abundant data from the internet and cheap sensors gave models enough examples to learn from; fast graphics chips gave the computing power to train large networks; and free, efficient frameworks made building and training whole systems easy for many people.