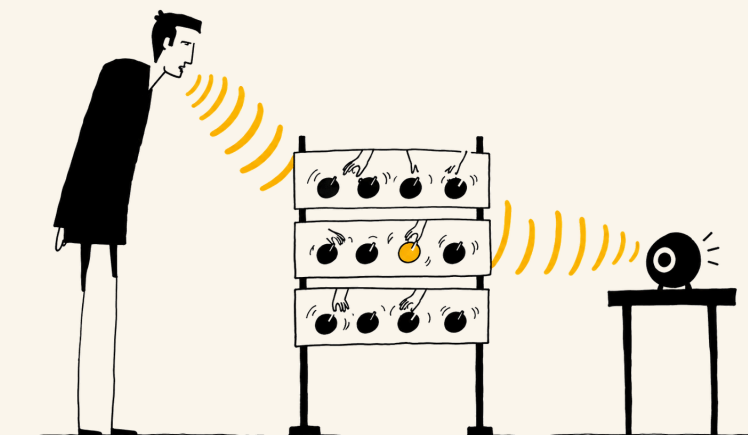


Dive into Deep Learning



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

Introduction

A Motivating Example

Key Components

Models and Objective Functions

Optimization Algorithms and Supervised Learning

Regression

Classification

Tagging, Search and Recommender Systems

Sequence Learning

Unsupervised and Self-Supervised Learning

Interacting with an Environment and Reinforcement Learning

Roots

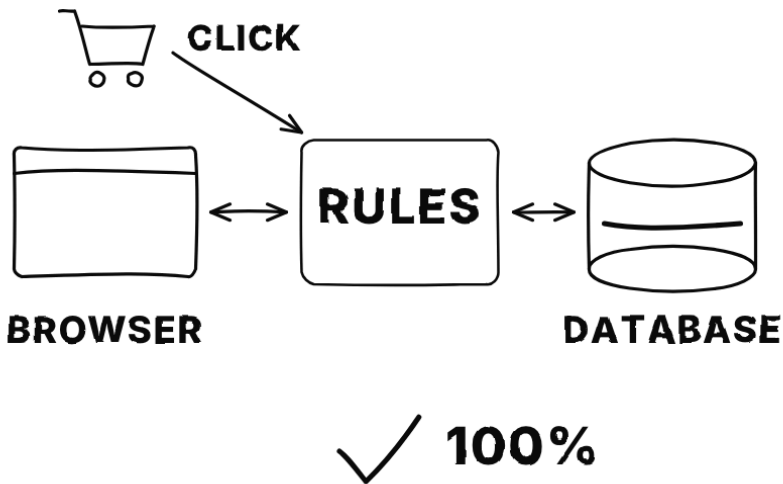
The Road to Deep Learning

Success Stories

The Essence of Deep Learning

Introduction

Until recently, nearly every computer program that you might have interacted with during an ordinary day was coded up as a rigid set of rules specifying precisely how it should behave. Say that we wanted to write an application to manage an e-commerce platform. After huddling around a whiteboard for a few hours to ponder the problem, we might settle on the broad strokes of a working solution, for example: first, users interact with the application through an interface running in a web browser or mobile application; second, our application interacts with a commercial-grade database engine to keep track of each user's state and maintain records of historical transactions; and third, at the heart of our application, the business logic (you might say, the brains) of our application spells out a set of rules that map every conceivable circumstance to the corresponding action that our program should take.

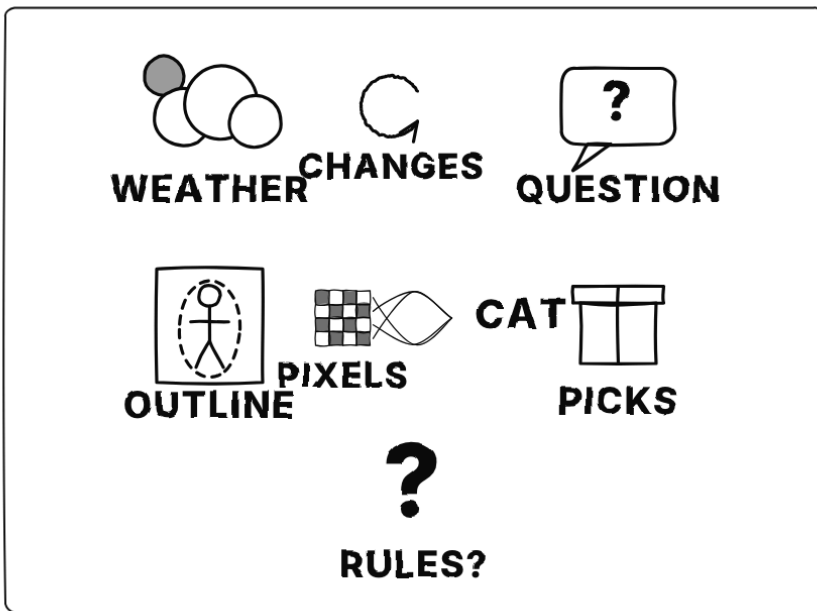


A shop built from rules

To build the brains of our application, we might enumerate all the common events that our program should handle. For example, whenever a customer clicks to add an item to their shopping cart, our program should add an entry to the shopping cart database table, associating that user's ID with the requested product's ID. We might then attempt to step through every possible corner case, testing the appropriateness of our rules and making any necessary modifications. What happens if a user initiates a purchase with an empty cart? While few developers ever get it completely right the first time (it might take some test runs to work out the kinks), for the most part we can write such programs and confidently launch them before ever seeing a real customer. Our ability to manually design automated systems that drive functioning products and systems, often in novel situations, is a

remarkable cognitive feat. And when you are able to devise solutions that work 100% of the time, you typically should not be worrying about machine learning.

Fortunately for the growing community of machine learning scientists, many tasks that we would like to automate do not bend so easily to human ingenuity. Imagine huddling around the whiteboard with the smartest minds you know, but this time you are tackling one of the following problems.



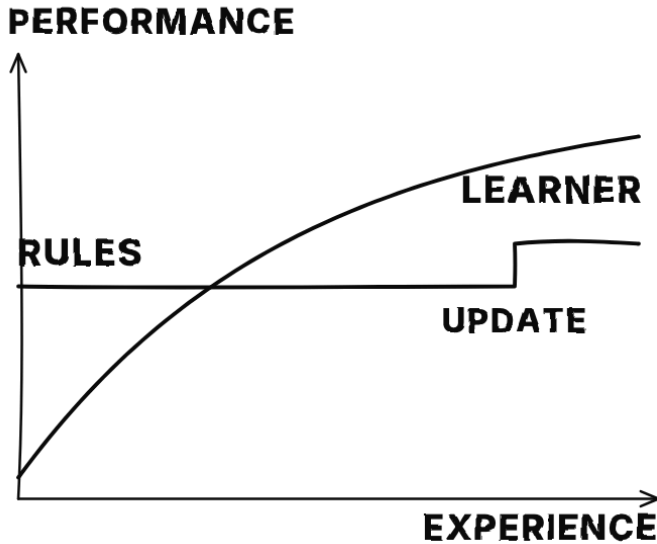
Tasks rules cannot write

Write a program that predicts tomorrow's weather given geographic information, satellite images, and a trailing window of past weather.
Write a program that takes in a factoid question, expressed in free-

form text, and answers it correctly. Write a program that, given an image, identifies every person depicted in it and draws outlines around each. Write a program that presents users with products that they are likely to enjoy but unlikely, in the natural course of browsing, to encounter.

For these problems, even elite programmers would struggle to code up solutions from scratch. The reasons can vary. Sometimes the program that we are looking for follows a pattern that changes over time, so there is no fixed right answer! In such cases, any successful solution must adapt gracefully to a changing world. At other times, the relationship (say between pixels, and abstract categories) may be too complicated, requiring thousands or millions of computations and following unknown principles. In the case of image recognition, the precise steps required to perform the task lie beyond our conscious understanding, even though our subconscious cognitive processes execute the task effortlessly.

Machine learning is the study of algorithms that can learn from experience. As a machine learning algorithm accumulates more experience, typically in the form of observational data or interactions with an environment, its performance improves. Contrast this with our deterministic e-commerce platform, which follows the same business logic, no matter how much experience accrues, until the developers themselves learn and decide that it is time to update the software. In this book, we will teach you the fundamentals of machine learning, focusing in particular on deep learning, a powerful set of techniques driving innovations in areas as diverse as computer vision, natural language processing, healthcare, and genomics.



Learning improves with experience

Questions

1. According to the passage, when should you typically not be worrying about machine learning?
 - a. When the program will run in a web browser
 - b. When you can devise rules that work 100% of the time
 - c. When the program must keep records of transactions
 - d. When the developers have not yet met a real customer

2. Why would a fixed set of rules fail to recommend products shoppers will enjoy, according to the passage's first reason?
 - a. Because the database cannot store enough products
 - b. Because the program has to run on phones as well as computers
 - c. Because the pattern may change over time, so there is no fixed right answer
 - d. Because shoppers never click on recommended products
3. Why is image recognition hard to program by hand, even though people do it effortlessly?
 - a. The precise steps lie beyond our conscious understanding
 - b. Pictures take up too much memory to store
 - c. Only a few people can recognise faces reliably
 - d. Images change too slowly to learn from
4. What distinguishes a machine learning algorithm from the e-commerce platform described in the passage?
 - a. It has no rules at all inside it
 - b. It runs faster on the same computer
 - c. It never needs developers to build it
 - d. Its performance improves as it accumulates experience

Answers

1. **b)** When you are able to devise solutions that work 100% of the time, you typically should not be worrying about machine learning.
2. **c)** Sometimes the pattern we are looking for changes over time, so any successful solution must adapt gracefully to a changing world.

3. a) The precise steps required lie beyond our conscious understanding, even though our subconscious processes execute the task effortlessly.

4. d) A machine learning algorithm improves as it accumulates experience, while the platform follows the same business logic until the developers update it.

Practice

1. Take the problem of predicting tomorrow's weather from geographic information, satellite images and past weather. Using the two reasons the passage gives, explain why programmers would struggle to write it as a fixed set of rules.

Hint: Ask whether the pattern stays the same over time, and how complicated the link from images to a forecast is.

Answer: The relationship between satellite images, past weather and tomorrow's weather is far too complicated to spell out, needing a huge number of computations that follow principles nobody can write down as rules. And patterns may shift over time, so a fixed answer written once would not keep up; a successful solution must adapt to a changing world.

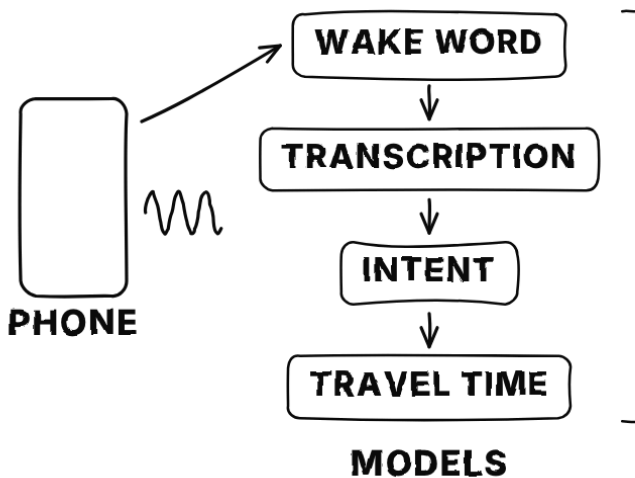
2. In your own words, explain the difference between how the e-commerce platform improves and how a machine learning algorithm improves.

Hint: Ask who does the learning in each case.

Answer: The platform follows the same business logic however much experience accrues; it changes only when the developers themselves learn something and update the software. A machine learning algorithm improves on its own as it gathers more experience, such as observational data or interactions with an environment.

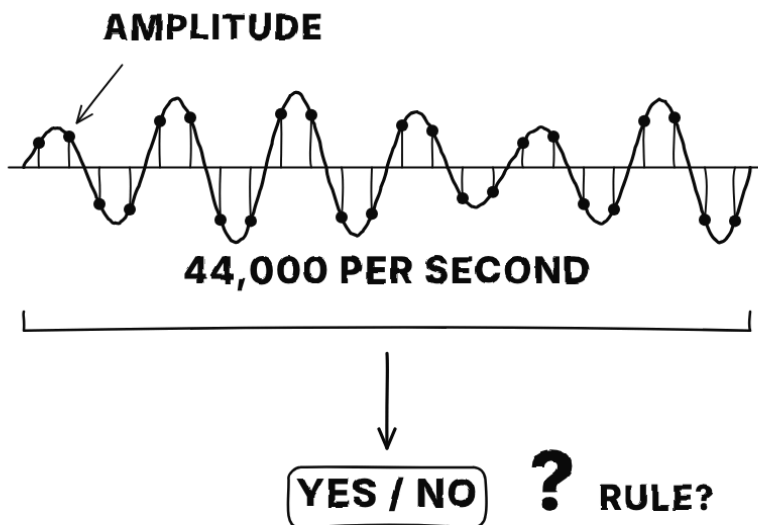
A Motivating Example

Before beginning writing, the authors of this book, like much of the work force, had to become caffeinated. We hopped in the car and started driving. Using an iPhone, Alex called out "Hey Siri", awakening the phone's voice recognition system. Then Mu commanded "directions to Blue Bottle coffee shop". The phone quickly displayed the transcription of his command. It also recognized that we were asking for directions and launched the Maps application (app) to fulfill our request. Once launched, the Maps app identified a number of routes. Next to each route, the phone displayed a predicted transit time. While this story was fabricated for pedagogical convenience, it demonstrates that in the span of just a few seconds, our everyday interactions with a smart phone can engage several machine learning models.



Several models in one phone

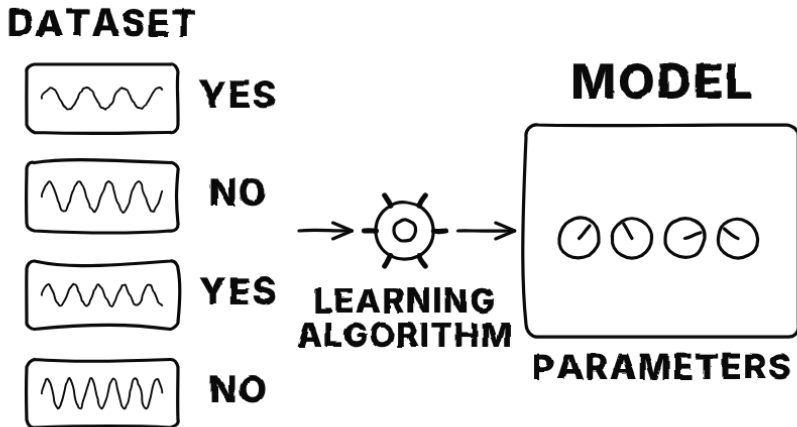
Imagine just writing a program to respond to a wake word such as "Alexa", "OK Google", and "Hey Siri". Try coding it up in a room by yourself with nothing but a computer and a code editor, as illustrated in the figure. How would you write such a program from first principles? Think about it... the problem is hard. Every second, the microphone will collect roughly 44,000 samples. Each sample is a measurement of the amplitude of the sound wave. What rule could map reliably from a snippet of raw audio to confident predictions, yes or no, about whether the snippet contains the wake word? If you are stuck, do not worry. We do not know how to write such a program from scratch either. That is why we use machine learning.



Raw audio to yes or no

Here is the trick. Often, even when we do not know how to tell a computer explicitly how to map from inputs to outputs, we are nonetheless capable of performing the cognitive feat ourselves. In other words, even if you do not know how to program a computer to recognize the word "Alexa", you yourself are able to recognize it. Armed with this ability, we can collect a huge dataset containing examples of audio snippets and associated labels, indicating which snippets contain the wake word. In the currently dominant approach to machine learning, we do not attempt to design a system explicitly to recognize wake words. Instead, we define a flexible program whose behavior is determined by a number of parameters. Then we use the dataset to determine the best possible parameter values, that is, those

that improve the performance of our program with respect to a chosen performance measure.

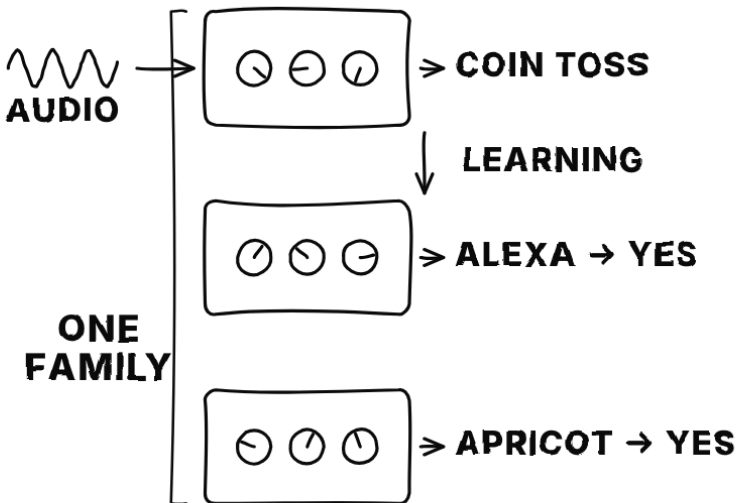


Dataset, knobs, learning algorithm

You can think of the parameters as knobs that we can turn, manipulating the behavior of the program. Once the parameters are fixed, we call the program a model. The set of all distinct programs (input–output mappings) that we can produce just by manipulating the parameters is called a family of models. And the "meta-program" that uses our dataset to choose the parameters is called a learning algorithm.

Before we can go ahead and engage the learning algorithm, we have to define the problem precisely, pinning down the exact nature of the

inputs and outputs, and choosing an appropriate model family. In this case, our model receives a snippet of audio as input, and the model generates a selection between yes and no as output. If all goes according to plan the model's guesses will typically be correct as to whether the snippet contains the wake word.

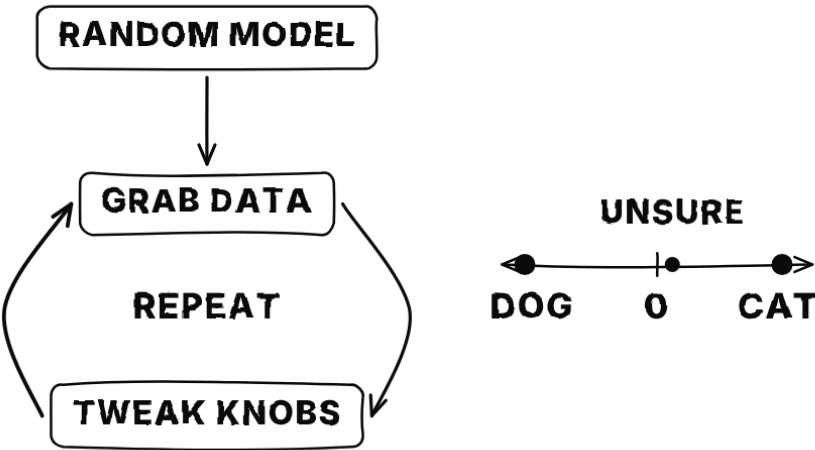


One family, many settings

If we choose the right family of models, there should exist one setting of the knobs such that the model fires "yes" every time it hears the word "Alexa". Because the exact choice of the wake word is arbitrary, we will probably need a model family sufficiently rich that, via another setting of the knobs, it could fire "yes" only upon hearing the word "Apricot". We expect that the same model family should be suitable for "Alexa" recognition and "Apricot" recognition because

they seem, intuitively, to be similar tasks. However, we might need a different family of models entirely if we want to deal with fundamentally different inputs or outputs, say if we wanted to map from images to captions, or from English sentences to Chinese sentences.

As you might guess, if we just set all of the knobs randomly, it is unlikely that our model will recognize "Alexa", "Apricot", or any other English word. In machine learning, the learning is the process by which we discover the right setting of the knobs for coercing the desired behavior from our model. In other words, we train our model with data. As shown in the figure, the training process usually looks like the following.



Training loop, then a cat score

First, start off with a randomly initialized model that cannot do anything useful. Second, grab some of your data (for example, audio snippets and corresponding yes or no labels). Third, tweak the knobs to make the model perform better as assessed on those examples. Fourth, repeat steps two and three until the model is awesome.

To summarize, rather than code up a wake word recognizer, we code up a program that can learn to recognize wake words, if presented with a large labeled dataset. You can think of this act of determining a program's behavior by presenting it with a dataset as programming with data. That is to say, we can "program" a cat detector by providing our machine learning system with many examples of cats and dogs. This way the detector will eventually learn to emit a very large positive number if it is a cat, a very large negative number if it is a dog, and something closer to zero if it is not sure. This barely scratches the surface of what machine learning can do. Deep learning, which we will explain in greater detail later, is just one among many popular methods for solving machine learning problems.

Questions

1. Why, according to the passage, can we build a dataset for wake-word recognition even though we cannot write the program?
 - a. Because microphones label the audio automatically
 - b. Because we ourselves can recognize the wake word and label snippets
 - c. Because wake words have a fixed shape in the sound wave
 - d. Because the phone already contains a list of all words

2. What turns a flexible program into what the passage calls a model?
 - a. Collecting a dataset for it
 - b. Choosing a performance measure
 - c. Fixing the values of its parameters
 - d. Running it on a smartphone
3. Why might the same family of models serve for both "Alexa" and "Apricot" but not for turning images into captions?
 - a. Captions need far more data than words
 - b. Images cannot be stored as numbers
 - c. Apricot is easier to say than Alexa
 - d. Image captioning has fundamentally different inputs and outputs
4. In the training process the passage describes, what happens after tweaking the knobs on some examples?
 - a. Grab more data and tweak again, until the model is good
 - b. Set all the knobs randomly again
 - c. Fix the parameters for good and stop
 - d. Replace the model family with a richer one

Answers

1. **b)** Even if you do not know how to program a computer to recognize the word, you yourself are able to recognize it, so you can collect audio snippets with labels.
2. **c)** The parameters are knobs; once the parameters are fixed, we call the program a model.

3. d) The two wake words are similar tasks, but a different family may be needed for fundamentally different inputs or outputs, such as images to captions.

4. a) The steps are: start from a random model, grab data, tweak the knobs to do better on it, and repeat steps two and three until the model is awesome.

Worked examples

How much the microphone hears The microphone collects roughly 44,000 samples every second. A wake-word model is given a snippet of 2 seconds of raw audio. How many amplitude measurements does it receive as input?

Each sample is one measurement of the amplitude of the sound wave, so we count samples.

The microphone takes about 44,000 samples in one second.

In 2 seconds it takes $2 \times 44,000 = 88,000$ samples.

The model must map all of these numbers to a single answer, yes or no, which is why no simple rule will do.

Answer: About 88,000 amplitude measurements

Practice

1. The cat detector emits a very large positive number for a cat and a very large negative number for a dog. What should it emit for a blurry photo it is unsure about, and why?

Hint: Think of the two ends of the scale and where doubt lies between them.

Answer: Something close to zero: large positive means cat, large negative means dog, and a value near zero is the middle of the scale, where the detector is not sure.

2. Explain in your own words what the passage means by "programming with data".

Hint: Compare writing the rules yourself with showing the program examples.

Answer: Instead of writing the rules of a wake-word recognizer or a cat detector ourselves, we write a flexible program and show it a large labeled dataset; a learning algorithm then sets its parameters. The program's behavior is determined by the examples we present, so the data does the programming.

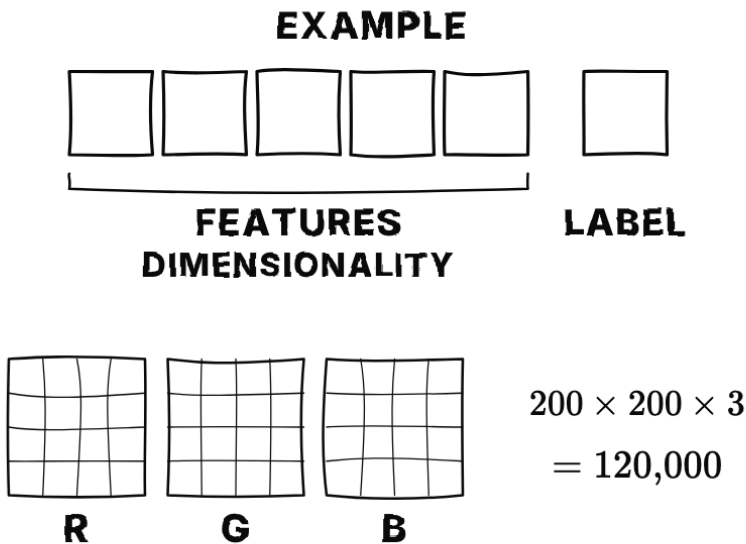
Key Components

In our wake word example, we described a dataset consisting of audio snippets and binary labels, and we gave a hand-wavy sense of how we might train a model to approximate a mapping from snippets to classifications. This sort of problem, where we try to predict a designated unknown label based on known inputs given a dataset consisting of examples for which the labels are known, is called supervised learning. This is just one among many kinds of machine learning problems. Before we explore other varieties, we would like to shed more light on some core components that will follow us around, no matter what kind of machine learning problem we tackle.

First, the data that we can learn from. Second, a model of how to transform the data. Third, an objective function that quantifies how well (or badly) the model is doing. Fourth, an algorithm to adjust the model's parameters to optimize the objective function.

It might go without saying that you cannot do data science without data. We could lose hundreds of pages pondering what precisely data is, but for now, we will focus on the key properties of the datasets that we will be concerned with. Generally, we are concerned with a collection of examples. In order to work with data usefully, we typically need to come up with a suitable numerical representation. Each example (or data point, data instance, sample) typically consists of a set of attributes called features (sometimes called covariates or inputs), based on which the model must make its predictions. In supervised learning problems, our goal is to predict the value of a

special attribute, called the label (or target), that is not part of the model's input.



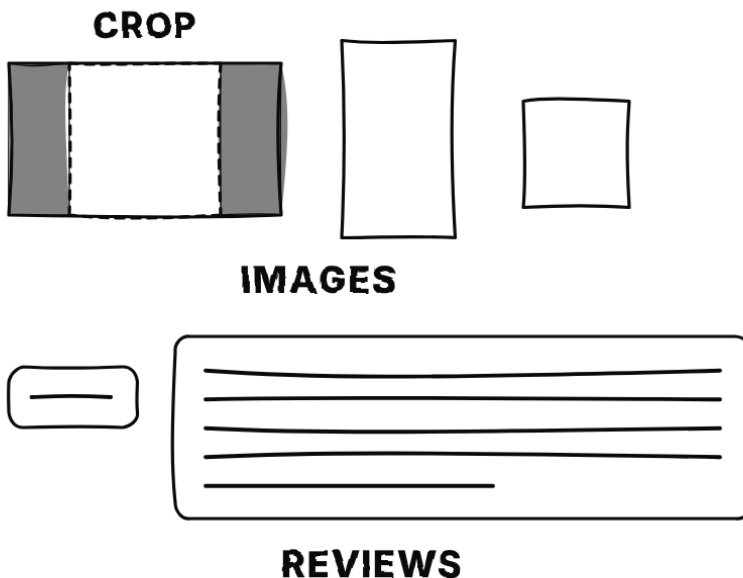
Features and a label

If we were working with image data, each example might consist of an individual photograph (the features) and a number indicating the category to which the photograph belongs (the label). The photograph would be represented numerically as three grids of numerical values representing the brightness of red, green, and blue light at each pixel location. For example, a 200×200 pixel color photograph would consist of $200 \times 200 \times 3 = 120,000$ numerical values.

Alternatively, we might work with electronic health record data and tackle the task of predicting the likelihood that a given patient will

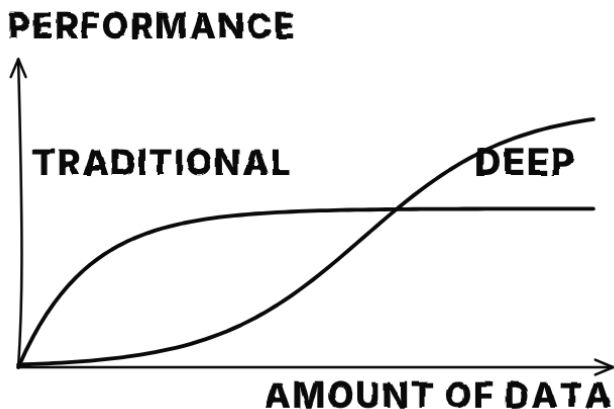
survive the next 30 days. Here, our features might consist of a collection of readily available attributes and frequently recorded measurements, including age, vital signs, comorbidities, current medications, and recent procedures. The label available for training would be a binary value indicating whether each patient in the historical data survived within the 30-day window.

In such cases, when every example is characterized by the same number of numerical features, we say that the inputs are fixed-length vectors and we call the (constant) length of the vectors the dimensionality of the data. As you might imagine, fixed-length inputs can be convenient, giving us one less complication to worry about. However, not all data can easily be represented as fixed-length vectors. While we might expect microscope images to come from standard equipment, we cannot expect images mined from the Internet all to have the same resolution or shape. For images, we might consider cropping them to a standard size, but that strategy only gets us so far. We risk losing information in the cropped-out portions. Moreover, text data resists fixed-length representations even more stubbornly. Consider the customer reviews left on e-commerce sites such as Amazon, IMDb, and TripAdvisor. Some are short: "it stinks!". Others ramble for pages. One major advantage of deep learning over traditional methods is the comparative grace with which modern models can handle varying-length data.



Data that will not line up

Generally, the more data we have, the easier our job becomes. When we have more data, we can train more powerful models and rely less heavily on preconceived assumptions. The regime change from (comparatively) small to big data is a major contributor to the success of modern deep learning. To drive the point home, many of the most exciting models in deep learning do not work without large datasets. Some others might work in the small data regime, but are no better than traditional approaches.

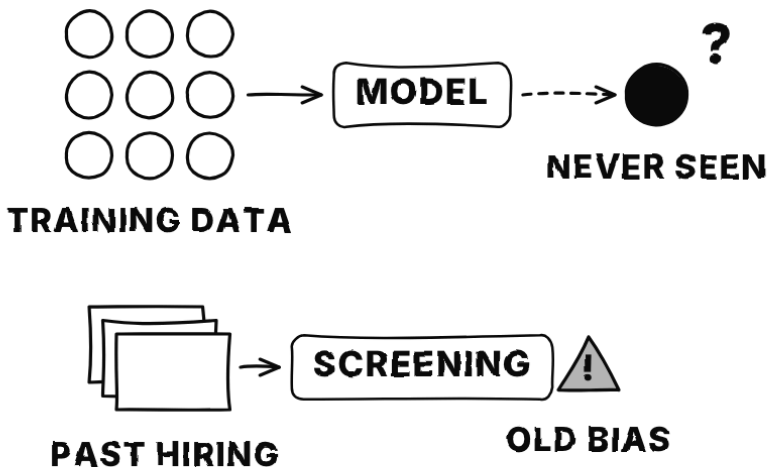


GARBAGE IN, GARBAGE OUT

More data, bigger models

Finally, it is not enough to have lots of data and to process it cleverly. We need the right data. If the data is full of mistakes, or if the chosen features are not predictive of the target quantity of interest, learning is going to fail. The situation is captured well by the cliché: garbage in, garbage out. Moreover, poor predictive performance is not the only potential consequence. In sensitive applications of machine learning, like predictive policing, resume screening, and risk models used for lending, we must be especially alert to the consequences of garbage data. One commonly occurring failure mode concerns datasets where some groups of people are unrepresented in the training data. Imagine applying a skin cancer recognition system that had never seen black skin before. Failure can also occur when the data does not only under-represent some groups but reflects societal prejudices. For

example, if past hiring decisions are used to train a predictive model that will be used to screen resumes then machine learning models could inadvertently capture and automate historical injustices. Note that this can all happen without the data scientist actively conspiring, or even being aware.



Missing groups, unfair results

Questions

1. In a supervised learning problem about photographs, which part of an example is the label?
 - a. The brightness values of the red, green and blue light

- b. The number indicating the category of the photograph
 - c. The width and height of the photograph in pixels
 - d. The grid of pixel locations
2. Why does cropping Internet images to a standard size only get us so far?
- a. Cropped images cannot be represented numerically
 - b. Cropping changes the colours of the pixels
 - c. We risk losing information in the cropped-out portions
 - d. Cropped images are always too small to learn from
3. Why is text even harder than images to represent as fixed-length vectors?
- a. Reviews range from a few words to many pages
 - b. Text contains no numerical information
 - c. Text is always shorter than an image
 - d. Reviews are written in many alphabets
4. A hiring model is trained on past hiring decisions that reflected prejudice. What does the passage warn may happen?
- a. The model will refuse to make predictions
 - b. The model will correct the past injustices
 - c. The model will only fail if the data scientist intends it
 - d. The model may capture and automate historical injustices

Answers

1. **b)** Each example might consist of a photograph, the features, and a number indicating the category to which it belongs, the label, which is not part of the model's input.

- 2. c)** Cropping gives fixed-length inputs, but we risk losing information in the cropped-out portions.
- 3. a)** Customer reviews vary enormously: some are as short as "it stinks!", others ramble for pages.
- 4. d)** If past hiring decisions are used to train a model that screens resumes, it could inadvertently capture and automate historical injustices, even without anyone being aware.

Worked examples

Counting the numbers in a photograph A color photograph is 200×200 pixels. Represented as three grids of brightness values for red, green and blue light, how many numerical values does it consist of?

Each grid has one value per pixel location: $200 \times 200 = 40,000$ values.

There are three grids, one each for red, green and blue light.

So the photograph is $40,000 \times 3 = 120,000$ numerical values.

Answer: 120,000 numerical values

Practice

1. Every example in a dataset of 100×100 pixel color photographs is a fixed-length vector. What is the dimensionality of the data?

Hint: Count the values per pixel and the number of pixels, as for the 200×200 photograph.

Answer: 30,000: there are $100 \times 100 = 10,000$ pixels, each with three values for red, green and blue, and the constant length of the vectors is the dimensionality.

2. For the task of predicting whether a patient will survive the next 30 days, name some features and the label, and say what kind of value

the label is.

Hint: The features are what the model sees; the label is what it must predict.

Answer: Features could be age, vital signs, comorbidities, current medications and recent procedures; the label is whether each patient in the historical data survived within the 30-day window, a binary value.

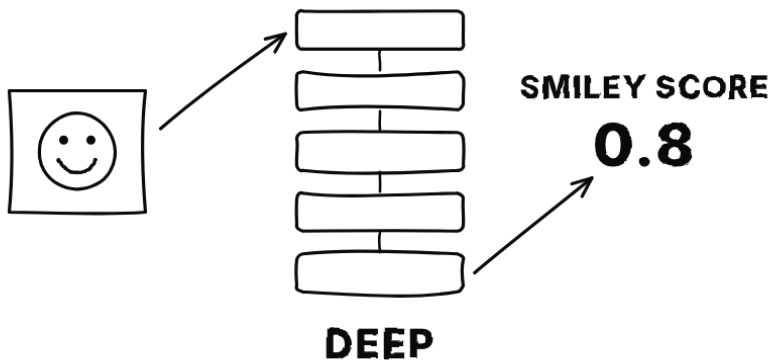
3. Explain in your own words what "garbage in, garbage out" means for machine learning, and give one consequence beyond poor predictions.

Hint: Think about mistaken data, unresponsive features, and groups missing from the data.

Answer: If the data is full of mistakes or the features do not predict the target, learning will fail however clever the processing. Beyond poor predictions, a system can fail people unfairly, for example a skin cancer recognizer that never saw black skin, or a resume screener that automates past prejudice.

Models and Objective Functions

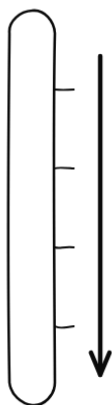
Most machine learning involves transforming the data in some sense. We might want to build a system that ingests photos and predicts smiley-ness. Alternatively, we might want to ingest a set of sensor readings and predict how normal versus anomalous the readings are. By model, we denote the computational machinery for ingesting data of one type, and spitting out predictions of a possibly different type. In particular, we are interested in statistical models that can be estimated from data. While simple models are perfectly capable of addressing appropriately simple problems, the problems that we focus on in this book stretch the limits of classical methods. Deep learning is differentiated from classical approaches principally by the set of powerful models that it focuses on. These models consist of many successive transformations of the data that are chained together top to bottom, thus the name deep learning. On our way to discussing deep models, we will also discuss some more traditional methods.



A model, shallow and deep

Earlier, we introduced machine learning as learning from experience. By learning here, we mean improving at some task over time. But who is to say what constitutes an improvement? You might imagine that we could propose updating our model, and some people might disagree on whether our proposal constituted an improvement or not.

LOSS



BETTER

score $\uparrow$

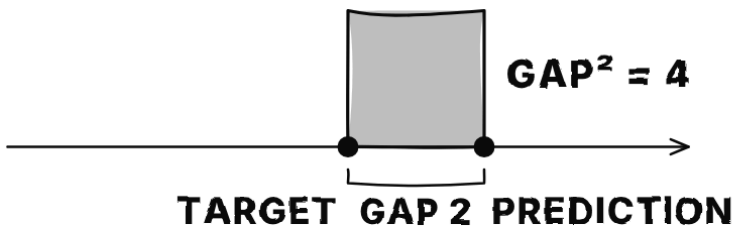
loss = - score $\downarrow$

Lower is better

In order to develop a formal mathematical system of learning machines, we need to have formal measures of how good (or bad) our models are. In machine learning, and optimization more generally, we call these objective functions. By convention, we usually define objective functions so that lower is better. This is merely a convention. You can take any function for which higher is better, and turn it into a new function that is qualitatively identical but for which lower is better by flipping the sign. Because we choose lower to be better, these functions are sometimes called loss functions.

When trying to predict numerical values, the most common loss function is squared error, that is, the square of the difference between the prediction and the ground truth target. For classification, the most

common objective is to minimize error rate, that is, the fraction of examples on which our predictions disagree with the ground truth. Some objectives (for example, squared error) are easy to optimize, while others (for example, error rate) are difficult to optimize directly, owing to non-differentiability or other complications. In these cases, it is common instead to optimize a surrogate objective.

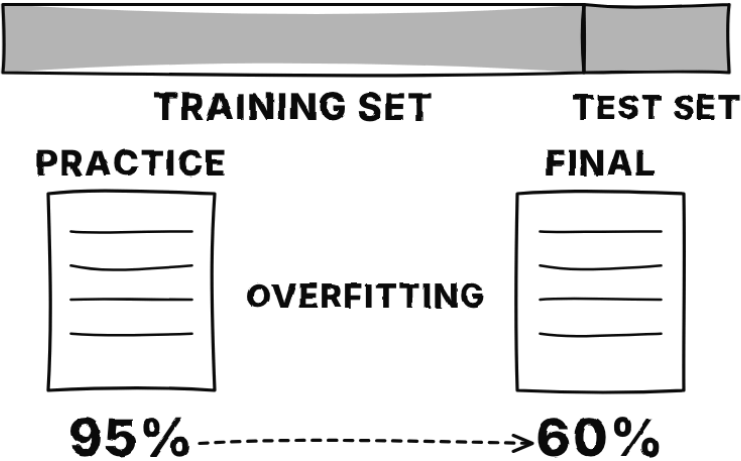


ERROR RATE 3/10
USE A SURROGATE

Squared error and error rate

During optimization, we think of the loss as a function of the model's parameters, and treat the training dataset as a constant. We learn the best values of our model's parameters by minimizing the loss incurred on a set consisting of some number of examples collected for training. However, doing well on the training data does not guarantee that we will do well on unseen data. So we will typically want to split the

available data into two partitions: the training dataset (or training set), for learning model parameters; and the test dataset (or test set), which is held out for evaluation. At the end of the day, we typically report how our models perform on both partitions. You could think of training performance as analogous to the scores that a student achieves on the practice exams used to prepare for some real final exam. Even if the results are encouraging, that does not guarantee success on the final exam. Over the course of studying, the student might begin to memorize the practice questions, appearing to master the topic but faltering when faced with previously unseen questions on the actual final exam. When a model performs well on the training set but fails to generalize to unseen data, we say that it is overfitting to the training data.



Train set, test set, overfitting

Questions

1. What, according to the passage, principally distinguishes deep learning from classical approaches?
 - a. It needs no objective function
 - b. Its powerful models chain many successive transformations of the data
 - c. It works only on photographs
 - d. It never uses statistical models
2. You have an objective where higher is better, such as accuracy. How can you make it fit the usual convention?
 - a. Square it
 - b. Divide it by the number of examples
 - c. Flip its sign, so that lower is better
 - d. Replace it with the error rate on the test set
3. Why do practitioners often optimize a surrogate objective instead of the error rate?
 - a. Error rate is difficult to optimize directly, owing to non-differentiability
 - b. Error rate can only be used for numerical predictions
 - c. Error rate is always higher than squared error
 - d. Error rate ignores the training data
4. A model scores very well on its training set but poorly on the held-out test set. What is this called?
 - a. Using a surrogate objective

- b. Flipping the sign of the loss
- c. Treating the dataset as a constant
- d. Overfitting to the training data

Answers

1. **b)** Deep learning is differentiated mainly by its powerful models, which consist of many successive transformations of the data chained together, hence the name.
2. **c)** Any function for which higher is better can be turned into a qualitatively identical one for which lower is better by flipping the sign.
3. **a)** Some objectives, such as error rate, are difficult to optimize directly, owing to non-differentiability or other complications, so a surrogate is optimized instead.
4. **d)** When a model performs well on the training set but fails to generalize to unseen data, it is overfitting, like a student who memorized the practice questions.

Worked examples

Squared error and error rate A model predicts a value of 7 where the ground truth target is 4. What is its squared error? Separately, a classifier gets 3 of 20 test examples wrong; what is its error rate?

Squared error is the square of the difference between prediction and target: $7 - 4 = 3$.

Squaring gives $3 \times 3 = 9$.

Error rate is the fraction of examples on which predictions disagree with the ground truth.

That is 3 out of 20, or $3 \div 20 = 0.15$.

Answer: Squared error 9; error rate 0.15 (15%)

Practice

1. A prediction of 10 is compared with a ground truth of 12. What is the squared error, and would a prediction of 14 have a larger or the same squared error?

Hint: Square the difference; the sign of the difference disappears when squared.

Answer: $10 - 12 = -2$, squared gives 4. A prediction of 14 differs by +2, which also gives 4, so the squared error is the same.

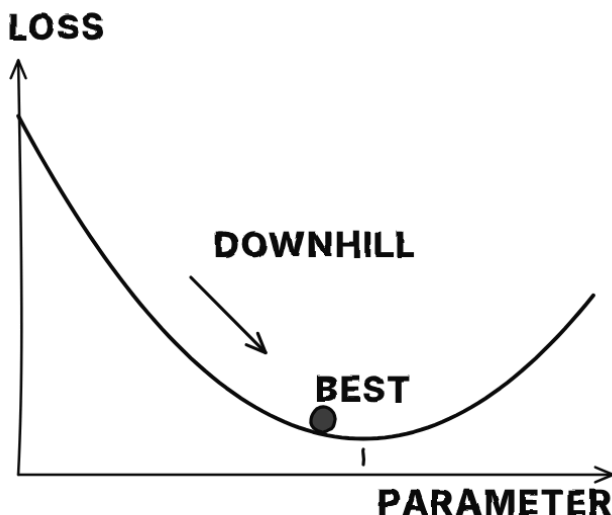
2. Explain in your own words why we split data into a training set and a test set, using the student and exam comparison.

Hint: What does a good score on the practice exams fail to prove?

Answer: Doing well on the training data does not guarantee doing well on unseen data. A student may memorize practice questions and then falter on new exam questions; likewise a model is trained on the training set and checked on a held-out test set, so we can see whether it generalizes rather than overfits.

Optimization Algorithms and Supervised Learning

Once we have got some data source and representation, a model, and a well-defined objective function, we need an algorithm capable of searching for the best possible parameters for minimizing the loss function. Popular optimization algorithms for deep learning are based on an approach called gradient descent. In brief, at each step, this method checks to see, for each parameter, how that training set loss would change if you perturbed that parameter by just a small amount. It would then update the parameter in the direction that lowers the loss.



Gradient descent, step by step

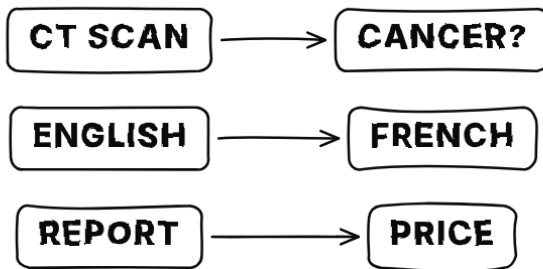
The wake word problem in our motivating example is just one among many that machine learning can tackle. To motivate the reader further and provide us with some common language that will follow us throughout the book, we now provide a broad overview of the landscape of machine learning problems.

Supervised learning describes tasks where we are given a dataset containing both features and labels and asked to produce a model that predicts the labels when given input features. Each feature–label pair is called an example. Sometimes, when the context is clear, we may use the term examples to refer to a collection of inputs, even when the corresponding labels are unknown. The supervision comes into play because, for choosing the parameters, we (the supervisors) provide

the model with a dataset consisting of labeled examples. In probabilistic terms, we typically are interested in estimating the conditional probability of a label given input features. While it is just one among several paradigms, supervised learning accounts for the majority of successful applications of machine learning in industry. Partly that is because many important tasks can be described crisply as estimating the probability of something unknown given a particular set of available data.



$$P(\text{label} \mid \text{features})$$

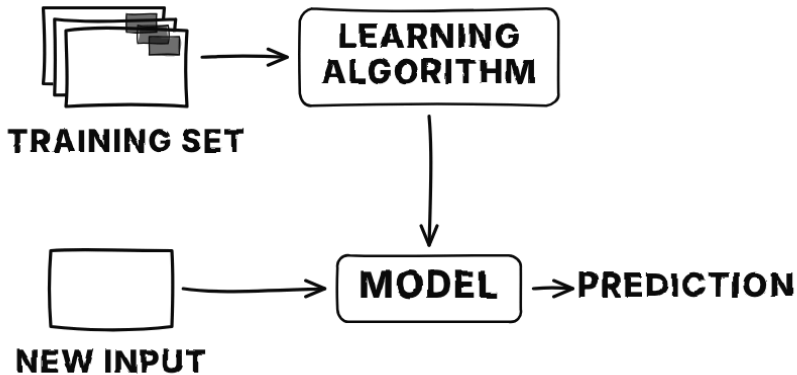


Predict labels from features

Predict cancer versus not cancer, given a computer tomography image. Predict the correct translation in French, given a sentence in English. Predict the price of a stock next month based on this month's financial reporting data.

While all supervised learning problems are captured by the simple description "predicting the labels given input features", supervised learning itself can take diverse forms and require tons of modeling decisions, depending on (among other considerations) the type, size, and quantity of the inputs and outputs. For example, we use different models for processing sequences of arbitrary lengths and fixed-length vector representations. We will visit many of these problems in depth throughout this book.

Informally, the learning process looks something like the following. First, grab a big collection of examples for which the features are known and select from them a random subset, acquiring the ground truth labels for each. Sometimes these labels might be available data that have already been collected (for example, did a patient die within the following year?) and other times we might need to employ human annotators to label the data (for example, assigning images to categories). Together, these inputs and corresponding labels comprise the training set. We feed the training dataset into a supervised learning algorithm, a function that takes as input a dataset and outputs another function: the learned model. Finally, we can feed previously unseen inputs to the learned model, using its outputs as predictions of the corresponding label. The full process is drawn in the figure.



From labelled data to a model

Questions

1. At each step of gradient descent, what does the method check for each parameter?
 - a. Whether the parameter is larger than all the others
 - b. How the training loss would change if the parameter were perturbed slightly
 - c. Whether the parameter appears in the test set
 - d. How many examples the parameter was trained on

2. Where does the "supervision" in supervised learning come from?
 - a. From an agent acting in an environment
 - b. From a test set held out for evaluation
 - c. From the labeled examples we provide for choosing the parameters
 - d. From the model checking its own answers
3. What does a supervised learning algorithm take as input and give as output, according to the passage?
 - a. It takes a dataset and outputs another function, the learned model
 - b. It takes a single input and outputs a label
 - c. It takes a model and outputs a dataset
 - d. It takes labels and outputs features

Answers

1. **b)** At each step the method checks, for each parameter, how the training set loss would change under a small perturbation, then updates it in the direction that lowers the loss.
2. **c)** We, the supervisors, provide the model with a dataset of labeled examples for choosing its parameters.
3. **a)** A supervised learning algorithm is a function that takes a dataset as input and outputs another function: the learned model.

Practice

1. For each task the passage lists, say what the input features are and what the label is: cancer detection, translation, and stock price

prediction.

Hint: Each task is phrased as "predict ..., given ...".

Answer: Cancer: features are a computer tomography image, label is cancer or not cancer. Translation: features are a sentence in English, label is its correct translation in French. Stock: features are this month's financial reporting data, label is the stock's price next month.

2. Describe in your own words the steps of the supervised learning process as the passage outlines it, including the two ways labels can be obtained.

Hint: Start from a big collection of examples whose features are known.

Answer: Take a big collection of examples with known features, pick a random subset and get the true labels for it, either from data already collected, such as whether a patient died within a year, or from human annotators who, for example, assign images to categories. These inputs and labels form the training set, which goes into a supervised learning algorithm that outputs a learned model. New, unseen inputs are then fed to the model, and its outputs are used as predictions of their labels.

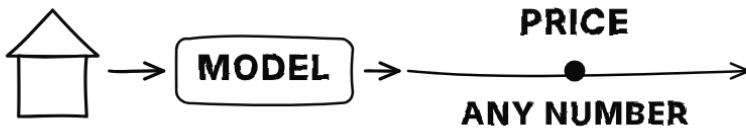
Regression

Perhaps the simplest supervised learning task to wrap your head around is regression. Consider, for example, a set of data harvested from a database of home sales. We might construct a table, in which each row corresponds to a different house, and each column corresponds to some relevant attribute, such as the square footage of a house, the number of bedrooms, the number of bathrooms, and the number of minutes (walking) to the center of town. In this dataset, each example would be a specific house, and the corresponding feature vector would be one row in the table. If you live in New York or San Francisco, and you are not the CEO of Amazon, Google, Microsoft, or Facebook, the (square footage, number of bedrooms, number of bathrooms, walking distance) feature vector for your home might look something like: [600, 1, 1, 60]. However, if you live in Pittsburgh, it might look more like [3000, 4, 3, 10]. Fixed-length feature vectors like this are essential for most classic machine learning algorithms.

ONE HOUSE				
	SQ FT	BEDS	BATHS	MIN
NEW YORK	600	1	1	60
PITTSBURGH	3000	4	3	10
FIXED LENGTH				

Houses as rows of numbers

What makes a problem a regression is actually the form of the target. Say that you are in the market for a new home. You might want to estimate the fair market value of a house, given some features such as above. The data here might consist of historical home listings and the labels might be the observed sales prices. When labels take on arbitrary numerical values (even within some interval), we call this a regression problem. The goal is to produce a model whose predictions closely approximate the actual label values.



HOW MUCH? HOW MANY?

MOVIE RATING

HOSPITAL DAYS

SURGERY HOURS

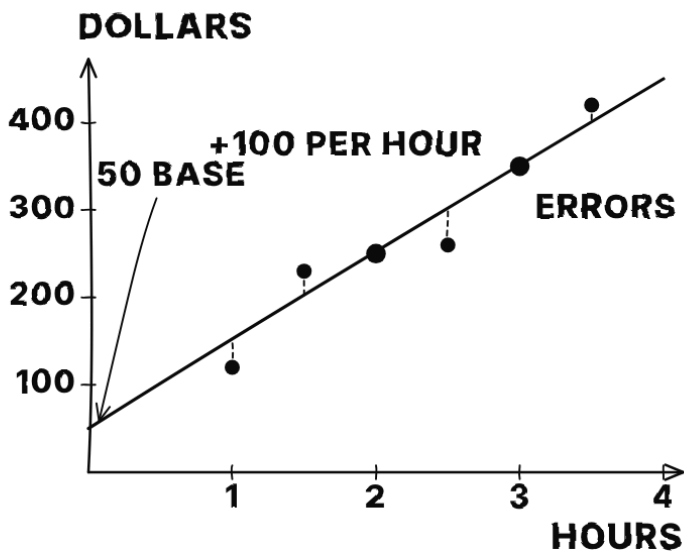
RAINFALL

Regression: the label is a number

Lots of practical problems are easily described as regression problems. Predicting the rating that a user will assign to a movie can be thought of as a regression problem and if you designed a great algorithm to accomplish this feat in 2009, you might have won the 1-million-dollar Netflix prize. Predicting the length of stay for patients in the hospital is also a regression problem. A good rule of thumb is that any how much? or how many? problem is likely to be regression. For example: how many hours will this surgery take? How much rainfall will this town have in the next six hours?

Even if you have never worked with machine learning before, you have probably worked through a regression problem informally. Imagine, for example, that you had your drains repaired and that your

contractor spent 3 hours removing gunk from your sewage pipes. Then they sent you a bill of 350 dollars. Now imagine that your friend hired the same contractor for 2 hours and received a bill of 250 dollars. If someone then asked you how much to expect on their upcoming gunk-removal invoice you might make some reasonable assumptions, such as more hours worked costs more dollars. You might also assume that there is some base charge and that the contractor then charges per hour. If these assumptions held true, then given these two data examples, you could already identify the contractor's pricing structure: 100 dollars per hour plus 50 dollars to show up at your house. If you followed that much, then you already understand the high-level idea behind linear regression.



The plumber's price line

In this case, we could produce the parameters that exactly matched the contractor's prices. Sometimes this is not possible, for example, if some of the variation arises from factors beyond your two features. In these cases, we will try to learn models that minimize the distance between our predictions and the observed values. In most of our chapters, we will focus on minimizing the squared error loss function. As we will see later, this loss corresponds to the assumption that our data were corrupted by Gaussian noise.

Questions

1. What, according to the passage, makes a problem a regression problem?
 - a. The features form a table with one row per example
 - b. The labels take on arbitrary numerical values
 - c. The model must be linear
 - d. The data comes from a database of sales
2. Which of these questions is most likely a regression problem, by the passage's rule of thumb?
 - a. Is this email spam or not?
 - b. Which of ten animals is in this photo?
 - c. How much rainfall will this town have in the next six hours?
 - d. Does this snippet contain the wake word?
3. When can we not find parameters that match the observed values exactly?

- a. When some of the variation arises from factors beyond our features
- b. When we have exactly two examples
- c. When the labels are dollar amounts
- d. When the contractor charges a base fee

Answers

- 1. **b)** What makes a problem a regression is the form of the target: when labels take on arbitrary numerical values, we call it regression.
- 2. **c)** A good rule of thumb is that any how much? or how many? problem is likely to be regression.
- 3. **a)** Exact matching is sometimes impossible, for example if some of the variation arises from factors beyond the features; then we minimize the distance between predictions and observations.

Worked examples

The contractor's pricing structure A contractor charges 350 dollars for 3 hours of work and 250 dollars for 2 hours. Assuming a base charge plus a fixed price per hour, what is the pricing structure, and what would 5 hours cost?

One extra hour (from 2 to 3 hours) adds $350 - 250 = 100$ dollars, so the hourly rate is 100 dollars.

For 2 hours the hourly part is $2 \times 100 = 200$ dollars, so the base charge is $250 - 200 = 50$ dollars.

Check with 3 hours: $50 + 3 \times 100 = 350$ dollars, as billed.

For 5 hours: $50 + 5 \times 100 = 550$ dollars.

Answer: 100 dollars per hour plus 50 dollars to show up; 5 hours would cost 550 dollars

Practice

1. A plumber bills 180 dollars for 2 hours and 300 dollars for 4 hours. Assuming a base charge plus a price per hour, find both.

Hint: Two extra hours add how much?

Answer: Two extra hours add $300 - 180 = 120$ dollars, so 60 dollars per hour; the base charge is $180 - 2 \times 60 = 60$ dollars.

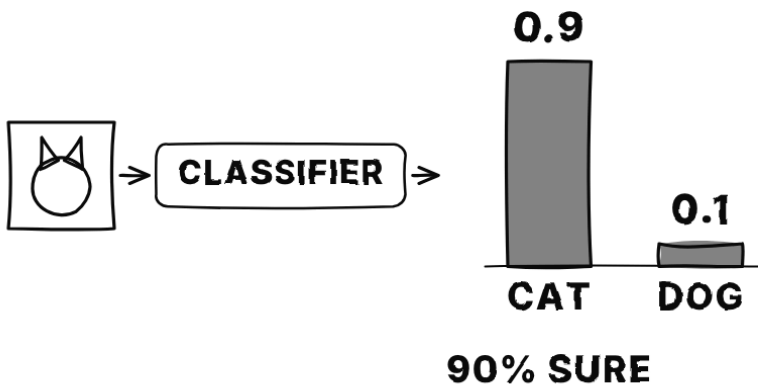
2. A house is described by square footage, bedrooms, bathrooms and walking minutes to the center. Write the feature vector of a 1,200 square-foot house with 2 bedrooms, 1 bathroom, 25 minutes from the center, and say what the label would be in a regression on home sales.

Hint: Keep the features in the passage's order.

Answer: The feature vector is $[1200, 2, 1, 25]$; the label would be the observed sales price, a numerical value.

Classification

While regression models are great for addressing how many? questions, lots of problems do not fit comfortably in this template. Consider, for example, a bank that wants to develop a check scanning feature for its mobile app. Ideally, the customer would simply snap a photo of a check and the app would automatically recognize the text from the image. Assuming that we had some ability to segment out image patches corresponding to each handwritten character, then the primary remaining task would be to determine which character among some known set is depicted in each image patch. These kinds of which one? problems are called classification and require a different set of tools from those used for regression, although many techniques will carry over.



LOSS: CROSS-ENTROPY

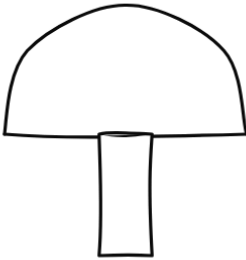
Which one? Probabilities per class

In classification, we want our model to look at features, for example, the pixel values in an image, and then predict to which category (sometimes called a class) among some discrete set of options, an example belongs. For handwritten digits, we might have ten classes, corresponding to the digits 0 through 9. The simplest form of classification is when there are only two classes, a problem which we call binary classification. For example, our dataset could consist of images of animals and our labels might be the classes cat and dog. Whereas in regression we sought a regressor to output a numerical value, in classification we seek a classifier, whose output is the predicted class assignment.

For reasons that we will get into as the book gets more technical, it can be difficult to optimize a model that can only output a firm categorical assignment, for example, either "cat" or "dog". In these cases, it is usually much easier to express our model in the language of probabilities. Given features of an example, our model assigns a probability to each possible class. Returning to our animal classification example where the classes are cat and dog, a classifier might see an image and output the probability that the image is a cat as 0.9. We can interpret this number by saying that the classifier is 90% sure that the image depicts a cat. The magnitude of the probability for the predicted class conveys a notion of uncertainty. It is not the only one available and we will discuss others in chapters dealing with more advanced topics.

When we have more than two possible classes, we call the problem multiclass classification. Common examples include handwritten character recognition, with the classes 0, 1, 2 and so on up to 9, then a, b, c, and so on. While we attacked regression problems by trying to minimize the squared error loss function, the common loss function for classification problems is called cross-entropy, whose name will be demystified when we introduce information theory in later chapters.

Note that the most likely class is not necessarily the one that you are going to use for your decision. Assume that you find a beautiful mushroom in your backyard as shown in the figure.



DEATH CAP 0.2
SAFE 0.8

EAT

$$0.2 \times \infty + 0.8 \times 0 = \infty$$

DISCARD ✓

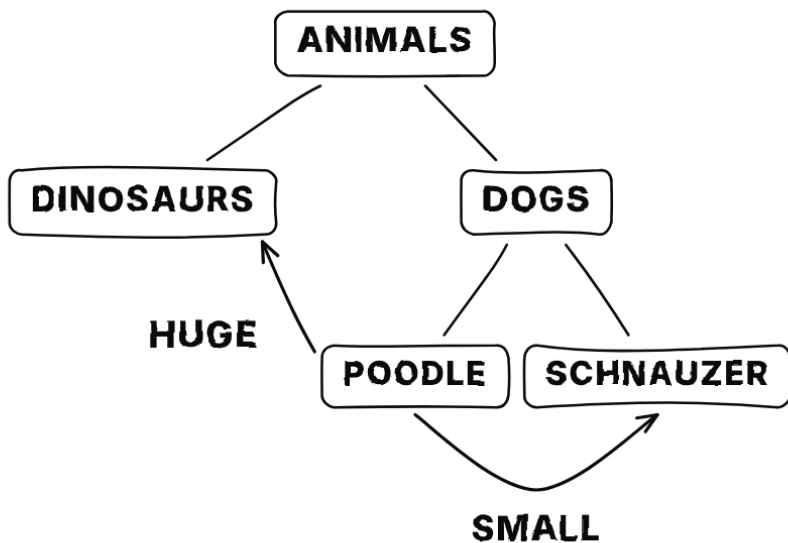
$$0.2 \times 0 + 0.8 \times 1 = 0.8$$

Expected loss of a mushroom

Now, assume that you built a classifier and trained it to predict whether a mushroom is poisonous based on a photograph. Say our poison-detection classifier outputs that the probability that the figure shows a death cap is 0.2. In other words, the classifier is 80% sure that our mushroom is not a death cap. Still, you would have to be a fool to eat it. That is because the certain benefit of a delicious dinner is not worth a 20% risk of dying from it. In other words, the effect of the uncertain risk outweighs the benefit by far. Thus, in order to make a decision about whether to eat the mushroom, we need to compute the expected detriment associated with each action which depends both on the likely outcomes and the benefits or harms associated with each. In this case, the detriment incurred by eating the mushroom might be $0.2 \times \infty + 0.8 \times 0 = \infty$, whereas the loss of discarding it is 0.2

$\times 0 + 0.8 \times 1 = 0.8$. Our caution was justified: as any mycologist would tell us, the mushroom in the figure is actually a death cap.

Classification can get much more complicated than just binary or multiclass classification. For instance, there are some variants of classification addressing hierarchically structured classes. In such cases not all errors are equal: if we must err, we might prefer to misclassify to a related class rather than a distant class. Usually, this is referred to as hierarchical classification. For inspiration, you might think of Linnaeus, who organized fauna in a hierarchy.



Near and far mistakes

In the case of animal classification, it might not be so bad to mistake a poodle for a schnauzer, but our model would pay a huge penalty if it

confused a poodle with a dinosaur. Which hierarchy is relevant might depend on how you plan to use the model. For example, rattlesnakes and garter snakes might be close on the phylogenetic tree, but mistaking a rattler for a garter could have fatal consequences.

Questions

1. Reading the handwritten characters of a check is classification rather than regression. Why?
 - a. Because the check is an image
 - b. Because the amount on the check is a number
 - c. Because each patch must be assigned to one character from a known set
 - d. Because the bank has many customers
2. Why do classifiers usually output a probability for each class rather than a firm answer?
 - a. A model that only outputs firm assignments can be difficult to optimize
 - b. Probabilities take less memory than class names
 - c. Firm answers are never correct
 - d. Probabilities remove all uncertainty
3. A classifier says a mushroom is a death cap with probability 0.2. Why should you still not eat it?
 - a. Because the classifier is wrong 80% of the time
 - b. Because 0.2 is the most likely class

- c. Because classifiers cannot judge mushrooms
 - d. Because the expected harm of eating outweighs the benefit of the dinner
4. In hierarchical classification, which mistake is the more costly one for a poodle?
- a. Calling it a schnauzer
 - b. Calling it a dinosaur
 - c. Both cost exactly the same
 - d. Neither, since only binary errors count

Answers

1. **c)** The remaining task is to determine which character among some known set is depicted in each patch: a which one? problem, called classification.
2. **a)** It can be difficult to optimize a model that can only output a firm categorical assignment, so it is easier to express the model in the language of probabilities.
3. **d)** The certain benefit of a dinner is not worth a 20% risk of dying; the expected detriment of eating is infinite, while discarding costs only 0.8.
4. **b)** Not all errors are equal: mistaking a poodle for a schnauzer might not be so bad, but confusing it with a dinosaur costs a huge penalty.

Worked examples

Expected detriment of the mushroom A classifier gives probability 0.2 that a mushroom is a death cap. Eating a death cap costs ∞ (death) and eating a safe mushroom costs 0; discarding a

death cap costs 0 and discarding a safe mushroom costs 1 (a lost dinner). Compute the expected detriment of each action.

The probability of a safe mushroom is $1 - 0.2 = 0.8$.

Eating: $0.2 \times \infty + 0.8 \times 0 = \infty$, since any chance of an infinite harm makes the total infinite.

Discarding: $0.2 \times 0 + 0.8 \times 1 = 0.8$.

The smaller expected detriment is discarding, so we throw the mushroom away.

Answer: Eating: ∞ ; discarding: 0.8. Discard it.

Practice

1. A different classifier says a mushroom is a death cap with probability 0.05. Using the same costs (eating a death cap ∞ , discarding a safe one 1), what are the expected detriments of eating and of discarding?

Hint: First find the probability that the mushroom is safe.

Answer: Safe with probability 0.95. Eating: $0.05 \times \infty + 0.95 \times 0 = \infty$.

Discarding: $0.05 \times 0 + 0.95 \times 1 = 0.95$. Discarding is still the better choice, because any risk of an infinite harm outweighs the dinner.

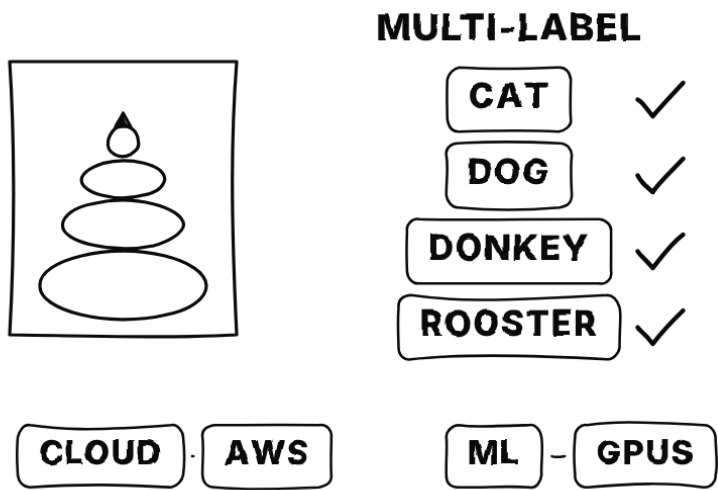
2. Say for each task whether it is binary or multiclass classification, and how many classes it has: cat or dog; handwritten digits; handwritten digits and lowercase letters a to z.

Hint: Count the possible categories.

Answer: Cat or dog: binary, 2 classes. Digits: multiclass, 10 classes (0 to 9). Digits and lowercase letters: multiclass, $10 + 26 = 36$ classes.

Tagging, Search and Recommender Systems

Some classification problems fit neatly into the binary or multiclass classification setups. For example, we could train a normal binary classifier to distinguish cats from dogs. Given the current state of computer vision, we can do this easily, with off-the-shelf tools. Nonetheless, no matter how accurate our model gets, we might find ourselves in trouble when the classifier encounters an image of the Town Musicians of Bremen, a popular German fairy tale featuring four animals, as in the figure.



Many labels at once

As you can see, the photo features a cat, a rooster, a dog, and a donkey, with some trees in the background. If we anticipate encountering such images, multiclass classification might not be the right problem formulation. Instead, we might want to give the model the option of saying the image depicts a cat, a dog, a donkey, and a rooster.

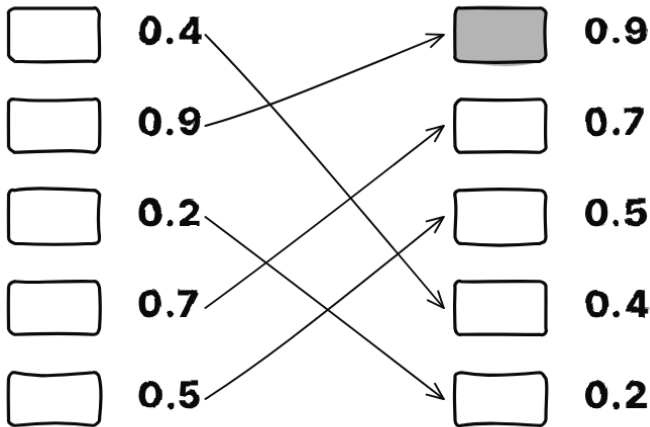
The problem of learning to predict classes that are not mutually exclusive is called multi-label classification. Auto-tagging problems are typically best described in terms of multi-label classification. Think of the tags people might apply to posts on a technical blog, for example, "machine learning", "technology", "gadgets", "programming languages", "Linux", "cloud computing", "AWS". A typical article might have 5 to 10 tags applied. Typically, tags will exhibit some correlation structure. Posts about "cloud computing" are likely to mention "AWS" and posts about "machine learning" are likely to mention "GPUs".

Sometimes such tagging problems draw on enormous label sets. The National Library of Medicine employs many professional annotators who associate each article to be indexed in PubMed with a set of tags drawn from the Medical Subject Headings (MeSH) ontology, a collection of roughly 28,000 tags. Correctly tagging articles is important because it allows researchers to conduct exhaustive reviews of the literature. This is a time-consuming process and typically there is a one-year lag between archiving and tagging. Machine learning can provide provisional tags until each article has a proper manual review. Indeed, for several years, the BioASQ organization has hosted competitions for this task.

In the field of information retrieval, we often impose ranks on sets of items. Take web search for example. The goal is less to determine whether a particular page is relevant for a query, but rather which, among a set of relevant results, should be shown most prominently to a particular user. One way of doing this might be to first assign a score to every element in the set and then to retrieve the top-rated elements. PageRank, the original secret sauce behind the Google search engine, was an early example of such a scoring system. Weirdly, the scoring provided by PageRank did not depend on the actual query. Instead, they relied on a simple relevance filter to identify the set of relevant candidates and then used PageRank to prioritize the more authoritative pages. Nowadays, search engines use machine learning and behavioral models to obtain query-dependent relevance scores. There are entire academic conferences devoted to this subject.

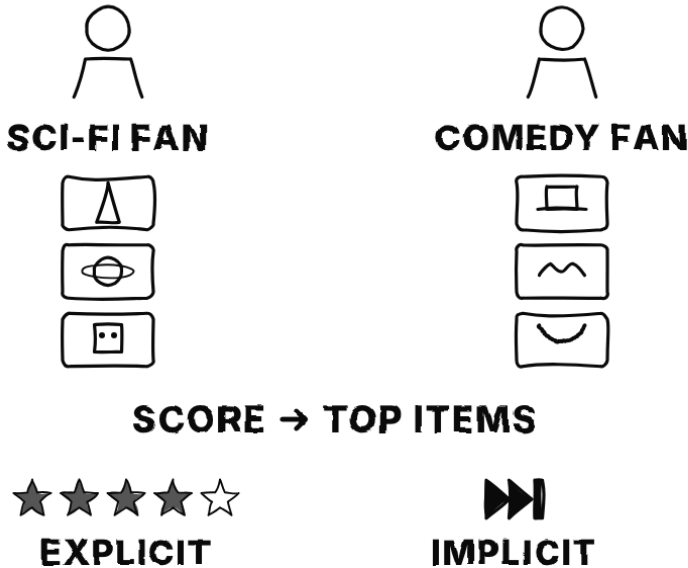
PAGERANK

SHOWN FIRST



Score, then rank

Recommender systems are another problem setting that is related to search and ranking. The problems are similar insofar as the goal is to display a set of items relevant to the user. The main difference is the emphasis on personalization to specific users in the context of recommender systems. For instance, for movie recommendations, the results page for a science fiction fan and the results page for a connoisseur of Peter Sellers comedies might differ significantly. Similar problems pop up in other recommendation settings, for example, for retail products, music, and news recommendation.



Same shop, different users

In some cases, customers provide explicit feedback, communicating how much they liked a particular product (for example, the product ratings and reviews on Amazon, IMDb, or Goodreads). In other cases, they provide implicit feedback, for example, by skipping titles on a playlist, which might indicate dissatisfaction or maybe just indicate that the song was inappropriate in context. In the simplest formulations, these systems are trained to estimate some score, such as an expected star rating or the probability that a given user will purchase a particular item.

Given such a model, for any given user, we could retrieve the set of objects with the largest scores, which could then be recommended to the user. Production systems are considerably more advanced and

take detailed user activity and item characteristics into account when computing such scores. The figure displays the deep learning books recommended by Amazon based on personalization algorithms tuned to capture Aston's preferences.

Despite their tremendous economic value, recommender systems naively built on top of predictive models suffer some serious conceptual flaws. To start, we only observe censored feedback: users preferentially rate movies that they feel strongly about. For example, on a five-point scale, you might notice that items receive many one- and five-star ratings but that there are conspicuously few three-star ratings. Moreover, current purchase habits are often a result of the recommendation algorithm currently in place, but learning algorithms do not always take this detail into account. Thus it is possible for feedback loops to form where a recommender system preferentially pushes an item that is then taken to be better (due to greater purchases) and in turn is recommended even more frequently. Many of these problems, about how to deal with censoring, incentives, and feedback loops, are important open research questions.



Censored ratings, feedback loops

Questions

1. Why is multiclass classification the wrong formulation for a photo of the Town Musicians of Bremen?
 - a. Because the photo has trees in the background
 - b. Because the animals are drawn, not photographed
 - c. Because several classes are present at once, and multiclass picks only one
 - d. Because there are too few classes

2. What was weird about how PageRank scored pages?
 - a. Its scores did not depend on the actual query
 - b. It scored pages only by their length
 - c. It ranked pages randomly
 - d. It used no relevance filter at all
3. What mainly distinguishes recommender systems from search and ranking?
 - a. They never use scores
 - b. They only work for movies
 - c. They show items that are not relevant
 - d. Their emphasis on personalization to specific users
4. How can a feedback loop form in a recommender system?
 - a. Users stop giving ratings altogether
 - b. An item pushed by the system is bought more, taken to be better, and recommended even more
 - c. The system forgets old purchases
 - d. Users rate every item three stars

Answers

1. **c)** The photo features a cat, a rooster, a dog and a donkey, so we want the model to be able to say all of them: the classes are not mutually exclusive.
2. **a)** The scoring provided by PageRank did not depend on the query; a simple relevance filter picked candidates and PageRank then prioritized the authoritative ones.
3. **d)** Both display relevant items; the main difference is the emphasis on personalization, so a science fiction fan and a Peter Sellers fan see different

pages.

4. b) Purchases partly result from the current recommendations; an item preferentially pushed gets more purchases, is taken to be better, and in turn is recommended even more.

Practice

1. Explain in your own words the difference between multiclass and multi-label classification, using the blog-tag example.

Hint: Ask how many classes one example may belong to at once.

Answer: In multiclass classification each example belongs to exactly one class out of several. In multi-label classification the classes are not mutually exclusive, so one example can have many; a blog post might carry 5 to 10 tags at once, such as cloud computing and AWS together.

2. Explain what "censored feedback" means for a movie recommender, and what pattern of star ratings it produces.

Hint: Who bothers to rate a movie?

Answer: Users mostly rate movies they feel strongly about, so the system sees only part of the opinions. On a five-point scale there are many one- and five-star ratings and conspicuously few three-star ones, so the ratings are not a fair picture of how people feel.

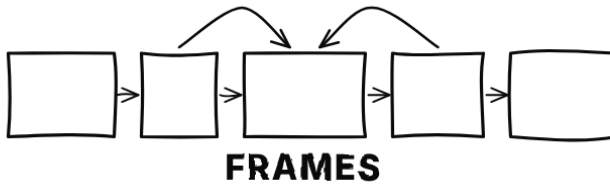
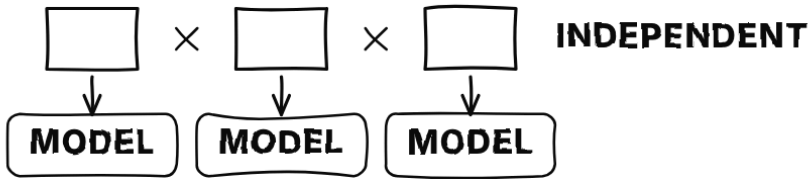
3. Give an example of explicit and of implicit feedback, and explain why implicit feedback can be ambiguous.

Hint: Think of ratings and of skipping a song.

Answer: Explicit feedback is a star rating or review, which says directly how much someone liked an item. Implicit feedback is behavior, such as skipping a song on a playlist; it may mean dissatisfaction, or only that the song did not fit the moment.

Sequence Learning

So far, we have looked at problems where we have some fixed number of inputs and produce a fixed number of outputs. For example, we considered predicting house prices given a fixed set of features: square footage, number of bedrooms, number of bathrooms, and the transit time to downtown. We also discussed mapping from an image (of fixed dimension) to the predicted probabilities that it belongs to each among a fixed number of classes and predicting star ratings associated with purchases based on the user ID and product ID alone. In these cases, once our model is trained, after each test example is fed into our model, it is immediately forgotten. We assumed that successive observations were independent and thus there was no need to hold on to this context.



SEQUENCE IN, SEQUENCE OUT

Independent inputs versus sequences

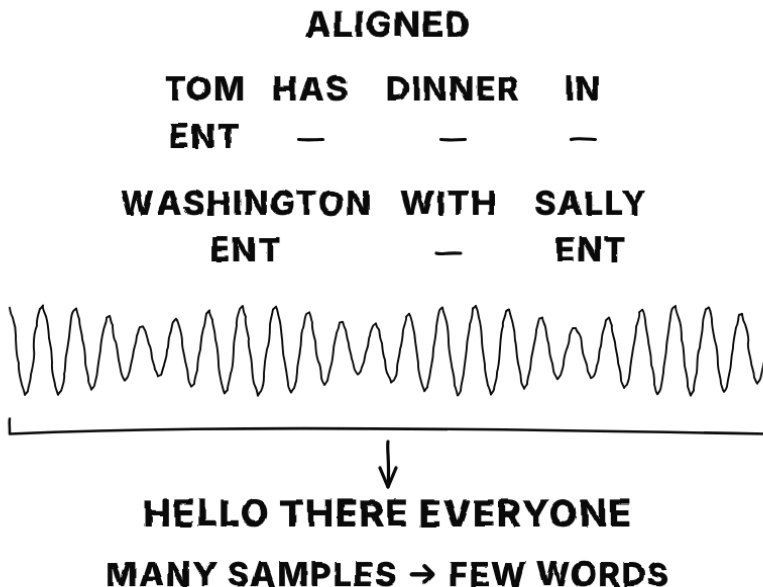
But how should we deal with video snippets? In this case, each snippet might consist of a different number of frames. And our guess of what is going on in each frame might be much stronger if we take into account the previous or succeeding frames. The same goes for language. For example, one popular deep learning problem is machine translation: the task of ingesting sentences in some source language and predicting their translations in another language.

Such problems also occur in medicine. We might want a model to monitor patients in the intensive care unit and to fire off alerts whenever their risk of dying in the next 24 hours exceeds some threshold. Here, we would not throw away everything that we know

about the patient history every hour, because we might not want to make predictions based only on the most recent measurements.

Questions like these are among the most exciting applications of machine learning and they are instances of sequence learning. They require a model either to ingest sequences of inputs or to emit sequences of outputs (or both). Specifically, sequence-to-sequence learning considers problems where both inputs and outputs consist of variable-length sequences. Examples include machine translation and speech-to-text transcription. While it is impossible to consider all types of sequence transformations, the following special cases are worth mentioning.

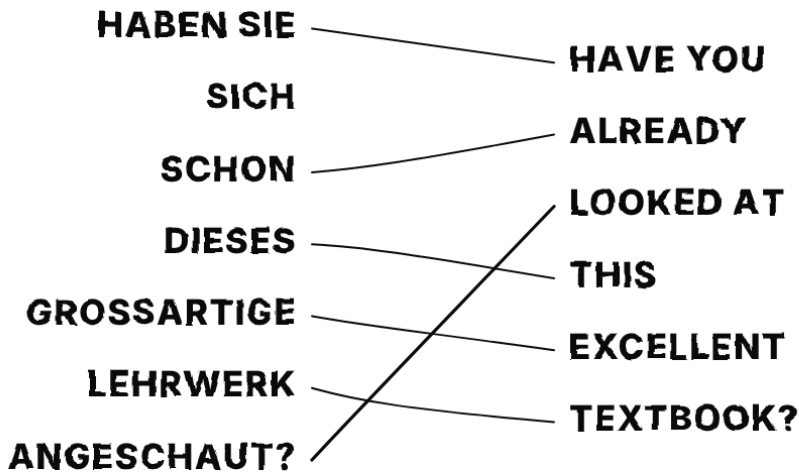
Tagging and parsing. This involves annotating a text sequence with attributes. Here, the inputs and outputs are aligned, that is, they are of the same number and occur in a corresponding order. For instance, in part-of-speech tagging, we annotate every word in a sentence with the corresponding part of speech, that is, "noun" or "direct object". Alternatively, we might want to know which groups of contiguous words refer to named entities, like people, places, or organizations. In the cartoonishly simple example below, we might just want to indicate whether or not any word in the sentence is part of a named entity (tagged as "Ent"). In the sentence "Tom has dinner in Washington with Sally", the words Tom, Washington and Sally are tagged "Ent", and every other word gets a dash.



Aligned tags, unaligned audio

Automatic speech recognition. With speech recognition, the input sequence is an audio recording of a speaker, as in the figure, and the output is a transcript of what the speaker said. The challenge is that there are many more audio frames (sound is typically sampled at 8 kHz or 16 kHz) than text, that is, there is no one-to-one correspondence between audio and text, since thousands of samples may correspond to a single spoken word. These are sequence-to-sequence learning problems, where the output is much shorter than the input. While humans are remarkably good at recognizing speech, even from low-quality audio, getting computers to perform the same feat is a formidable challenge.

Text to speech. This is the inverse of automatic speech recognition. Here, the input is text and the output is an audio file. In this case, the output is much longer than the input.



Translation reorders words

Machine translation. Unlike the case of speech recognition, where corresponding inputs and outputs occur in the same order, in machine translation, unaligned data poses a new challenge. Here the input and output sequences can have different lengths, and the corresponding regions of the respective sequences may appear in a different order. Consider the following illustrative example of the peculiar tendency of Germans to place the verbs at the end of sentences. In German: "Haben Sie sich schon dieses grossartige Lehrwerk angeschaut?" In English: "Have you already looked at this

excellent textbook?" With the wrong alignment: "Have you yourself already this excellent textbook looked at?"

Many related problems pop up in other learning tasks. For instance, determining the order in which a user reads a webpage is a two-dimensional layout analysis problem. Dialogue problems exhibit all kinds of additional complications, where determining what to say next requires taking into account real-world knowledge and the prior state of the conversation across long temporal distances. Such topics are active areas of research.

Questions

1. Why should a model watching patients in intensive care not forget the patient history every hour?
 - a. Because the hospital requires records to be kept
 - b. Because predictions based only on the latest measurements may miss what matters
 - c. Because each measurement is independent of the others
 - d. Because the model has a fixed number of inputs
2. In part-of-speech tagging, why are the inputs and outputs called aligned?
 - a. They are of the same number and occur in a corresponding order
 - b. They are both written in English
 - c. The output is much shorter than the input
 - d. They can appear in different orders

3. What makes speech recognition a sequence-to-sequence problem with no one-to-one correspondence?
 - a. The transcript is longer than the recording
 - b. Speech is always recorded in low quality
 - c. Thousands of audio samples may correspond to a single spoken word
 - d. Each audio frame corresponds to exactly one letter
4. What new challenge does machine translation pose that speech recognition does not?
 - a. The input is audio rather than text
 - b. The output is always shorter than the input
 - c. The sequences have a fixed length
 - d. Corresponding parts may appear in a different order

Answers

1. **b)** We would not throw away everything we know about the patient history, because we might not want to predict from the most recent measurements alone.
2. **a)** In tagging and parsing, inputs and outputs are aligned: one tag per word, in the same order as the words.
3. **c)** There are many more audio frames than text, since thousands of samples may correspond to a single word, so the output is much shorter than the input.
4. **d)** In machine translation the data is unaligned: sequences can differ in length and corresponding regions may appear in a different order, as with German verbs at the end.

Worked examples

How many samples in a word Sound is sampled at 16 kHz, that is, 16,000 samples per second. A speaker says one word in half a second. How many audio samples correspond to that single word?

16 kHz means 16,000 samples every second.

Half a second gives $16,000 \times 0.5 = 8,000$ samples.

All 8,000 samples correspond to one word of the transcript, which is why there is no one-to-one correspondence between audio and text.

Answer: 8,000 samples for one word

Practice

1. For each task, say whether the output is about as long as, much shorter than, or much longer than the input: part-of-speech tagging, speech recognition, text to speech.

Hint: Compare the number of items going in and coming out.

Answer: Part-of-speech tagging: as long as the input, one tag per word. Speech recognition: much shorter, thousands of samples become a few words. Text to speech: much longer, a few words become thousands of audio samples.

2. Tag the sentence "Sally visits Paris with Tom" in the passage's simple named-entity scheme, using "Ent" for words that are part of a named entity and a dash for the others.

Hint: People and places are named entities.

Answer: Sally: Ent, visits: dash, Paris: Ent, with: dash, Tom: Ent.

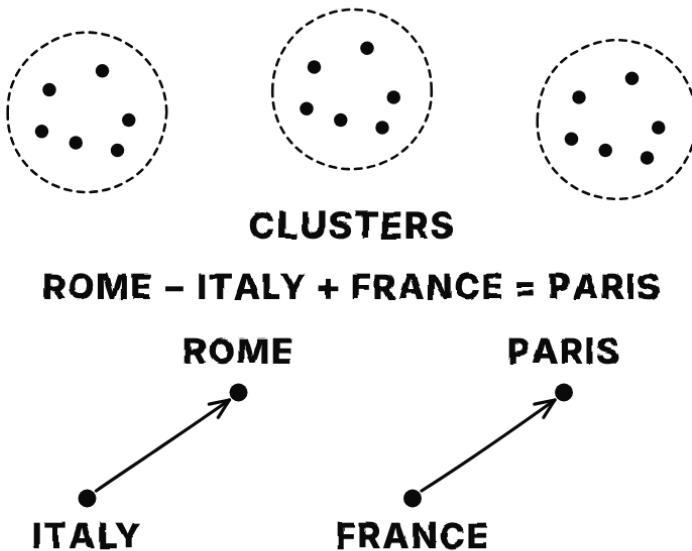
Unsupervised and Self-Supervised Learning

The previous examples focused on supervised learning, where we feed the model a giant dataset containing both the features and corresponding label values. You could think of the supervised learner as having an extremely specialized job and an extremely dictatorial boss. The boss stands over the learner's shoulder and tells them exactly what to do in every situation until they learn to map from situations to actions. Working for such a boss sounds pretty lame. On the other hand, pleasing such a boss is pretty easy. You just recognize the pattern as quickly as possible and imitate the boss's actions.

Considering the opposite situation, it could be frustrating to work for a boss who has no idea what they want you to do. However, if you plan to be a data scientist, you had better get used to it. The boss might just hand you a giant dump of data and tell you to do some data science with it! This sounds vague because it is vague. We call this class of problems unsupervised learning, and the type and number of questions we can ask is limited only by our creativity. We will address unsupervised learning techniques in later chapters. To whet your appetite for now, we describe a few of the following questions you might ask.

Can we find a small number of prototypes that accurately summarize the data? Given a set of photos, can we group them into landscape photos, pictures of dogs, babies, cats, and mountain peaks? Likewise, given a collection of users' browsing activities, can we group them into

users with similar behavior? This problem is typically known as clustering.



Clusters and word arithmetic

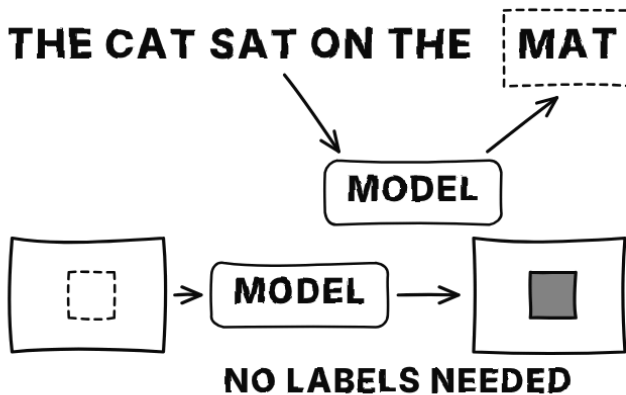
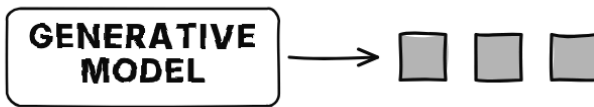
Can we find a small number of parameters that accurately capture the relevant properties of the data? The trajectories of a ball are well described by velocity, diameter, and mass of the ball. Tailors have developed a small number of parameters that describe human body shape fairly accurately for the purpose of fitting clothes. These problems are referred to as subspace estimation. If the dependence is linear, it is called principal component analysis.

Is there a representation of (arbitrarily structured) objects in Euclidean space such that symbolic properties can be well matched?

This can be used to describe entities and their relations, such as "Rome" – "Italy" + "France" = "Paris".

Is there a description of the root causes of much of the data that we observe? For instance, if we have demographic data about house prices, pollution, crime, location, education, and salaries, can we discover how they are related simply based on empirical data? The fields concerned with causality and probabilistic graphical models tackle such questions.

Another important and exciting recent development in unsupervised learning is the advent of deep generative models. These models estimate the density of the data, either explicitly or implicitly. Once trained, we can use a generative model either to score examples according to how likely they are, or to sample synthetic examples from the learned distribution. Early deep learning breakthroughs in generative modeling came with the invention of variational autoencoders and continued with the development of generative adversarial networks. More recent advances include normalizing flows and diffusion models.



Learning from the data itself

A further development in unsupervised learning has been the rise of self-supervised learning, techniques that leverage some aspect of the unlabeled data to provide supervision. For text, we can train models to "fill in the blanks" by predicting randomly masked words using their surrounding words (contexts) in big corpora without any labeling effort! For images, we may train models to tell the relative position between two cropped regions of the same image, to predict an occluded part of an image based on the remaining portions of the image, or to predict whether two examples are perturbed versions of the same underlying image. Self-supervised models often learn representations that are subsequently leveraged by fine-tuning the resulting models on some downstream task of interest.

Questions

1. In the boss analogy, what makes unsupervised learning different from supervised learning?
 - a. The boss checks every answer the learner gives
 - b. The learner gets no labels, only a dump of data and a vague request
 - c. The learner is told exactly what to do in every situation
 - d. The learner imitates the boss's actions
2. Grouping users by similar browsing behavior, with no labels given, is an example of what?
 - a. Clustering
 - b. Principal component analysis
 - c. Classification
 - d. Fine-tuning
3. What can a trained generative model be used for, according to the passage?
 - a. Only for clustering photos
 - b. Only for predicting labels
 - c. Scoring examples by how likely they are, or sampling synthetic examples
 - d. Discovering root causes of house prices
4. How does self-supervised learning get supervision without labeling effort?
 - a. It uses a dictatorial boss

- b. It asks human annotators for a few labels
- c. It only works on fully labeled data
- d. It uses some aspect of the unlabeled data itself, such as masked words

Answers

1. **b)** The unsupervised boss just hands over a giant dump of data and says do some data science with it; there are no labels telling what to do.
2. **a)** Finding a small number of prototypes, such as groups of users with similar behavior, is known as clustering.
3. **c)** Once trained, a generative model can score examples according to how likely they are, or sample synthetic examples from the learned distribution.
4. **d)** Self-supervised learning leverages the unlabeled data to provide supervision, for instance predicting randomly masked words from their contexts.

Practice

1. Explain in your own words what the example "Rome – Italy + France = Paris" shows about representing words in Euclidean space.

Hint: Think of what Rome is to Italy.

Answer: If words are placed as points in space so that relations match, then taking away Italy from Rome leaves something like "capital of", and adding France gives the capital of France, Paris. The representation captures symbolic relations between entities.

2. Describe two self-supervised tasks for images from the passage, and say where the supervision comes from.

Hint: The image itself supplies the answer.

Answer: One task is to tell the relative position of two cropped regions of the same image; another is to predict an occluded part from the rest of the image (or to tell whether two examples are perturbed versions of the same image). The answers come from the images themselves, so no human labels are needed.

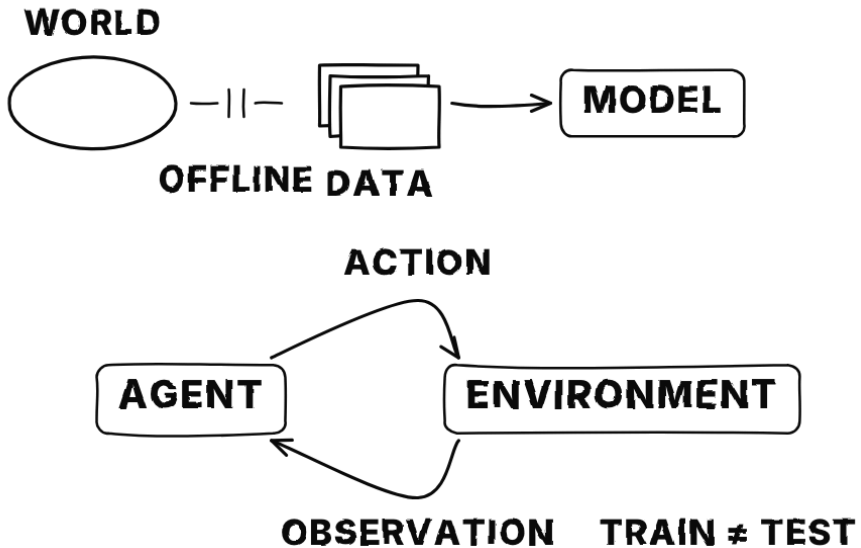
3. Explain the idea of subspace estimation using the tailor example.

Hint: How many numbers does a tailor need?

Answer: A body has countless details, but tailors use a small number of parameters, such as a few measurements, that describe body shape well enough to fit clothes. Subspace estimation looks for such a small set of parameters that captures the relevant properties of the data.

Interacting with an Environment and Reinforcement Learning

So far, we have not discussed where data actually comes from, or what actually happens when a machine learning model generates an output. That is because supervised learning and unsupervised learning do not address these issues in a very sophisticated way. In each case, we grab a big pile of data upfront, then set our pattern recognition machines in motion without ever interacting with the environment again. Because all the learning takes place after the algorithm is disconnected from the environment, this is sometimes called offline learning. For example, supervised learning assumes the simple interaction pattern depicted in the figure.

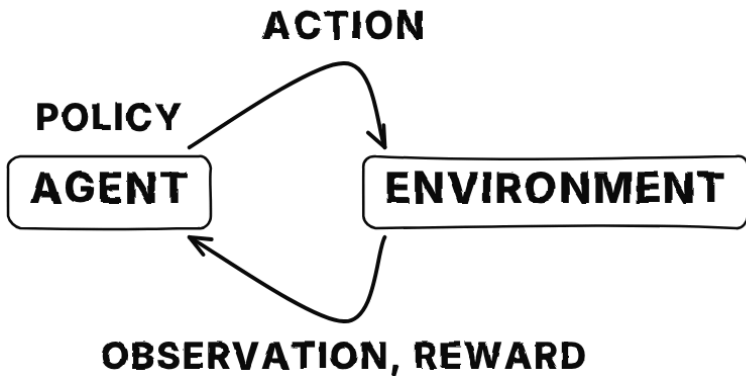


Offline learning versus acting

This simplicity of offline learning has its charms. The upside is that we can worry about pattern recognition in isolation, with no concern about complications arising from interactions with a dynamic environment. But this problem formulation is limiting. If you grew up reading Asimov's Robot novels, then you probably picture artificially intelligent agents capable not only of making predictions, but also of taking actions in the world. We want to think about intelligent agents, not just predictive models. This means that we need to think about choosing actions, not just making predictions. In contrast to mere predictions, actions actually impact the environment. If we want to train an intelligent agent, we must account for the way its actions might impact the future observations of the agent, and so offline learning is inappropriate.

Considering the interaction with an environment opens a whole set of new modeling questions. The following are just a few examples. Does the environment remember what we did previously? Does the environment want to help us, for example, a user reading text into a speech recognizer? Does the environment want to beat us, for example, spammers adapting their emails to evade spam filters? Does the environment have shifting dynamics? For example, would future data always resemble the past or would the patterns change over time, either naturally or in response to our automated tools?

These questions raise the problem of distribution shift, where training and test data are different. An example of this, that many of us may have met, is when taking exams written by a lecturer, while the homework was composed by their teaching assistants. Next, we briefly describe reinforcement learning, a rich framework for posing learning problems in which an agent interacts with an environment.



POLICY: OBSERVATION → ACTION

The reinforcement learning loop

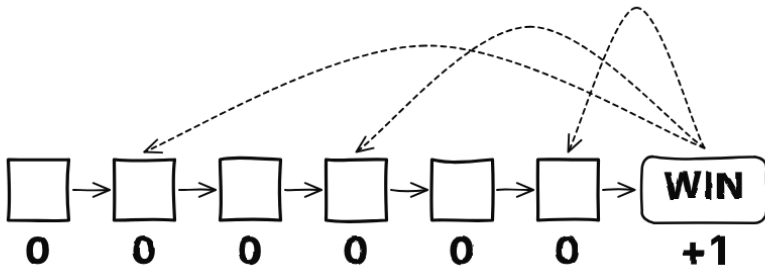
If you are interested in using machine learning to develop an agent that interacts with an environment and takes actions, then you are probably going to wind up focusing on reinforcement learning. This might include applications to robotics, to dialogue systems, and even to developing artificial intelligence (AI) for video games. Deep reinforcement learning, which applies deep learning to reinforcement learning problems, has surged in popularity. The breakthrough deep Q-network, that beat humans at Atari games using only the visual input, and the AlphaGo program, which dethroned the world champion at the board game Go, are two prominent examples.

Reinforcement learning gives a very general statement of a problem in which an agent interacts with an environment over a series of time

steps. At each time step, the agent receives some observation from the environment and must choose an action that is subsequently transmitted back to the environment via some mechanism (sometimes called an actuator), when, after each loop, the agent receives a reward from the environment. This process is illustrated in the figure. The agent then receives a subsequent observation, and chooses a subsequent action, and so on. The behavior of a reinforcement learning agent is governed by a policy. In brief, a policy is just a function that maps from observations of the environment to actions. The goal of reinforcement learning is to produce good policies.

It is hard to overstate the generality of the reinforcement learning framework. For example, supervised learning can be recast as reinforcement learning. Say we had a classification problem. We could create a reinforcement learning agent with one action corresponding to each class. We could then create an environment which gave a reward that was exactly equal to the loss function from the original supervised learning problem.

CREDIT?



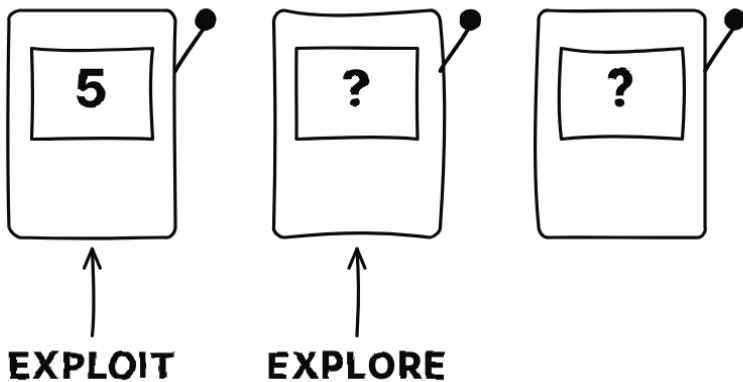
Reward only at the end

Further, reinforcement learning can also address many problems that supervised learning cannot. For example, in supervised learning, we always expect that the training input comes associated with the correct label. But in reinforcement learning, we do not assume that, for each observation the environment tells us the optimal action. In general, we just get some reward. Moreover, the environment may not even tell us which actions led to the reward.

Consider the game of chess. The only real reward signal comes at the end of the game when we either win, earning a reward of, say, 1, or when we lose, receiving a reward of, say, minus 1. So reinforcement learners must deal with the credit assignment problem: determining which actions to credit or blame for an outcome. The same goes for an

employee who gets a promotion on October 11. That promotion likely reflects a number of well-chosen actions over the previous year. Getting promoted in the future requires figuring out which actions along the way led to the earlier promotions.

Reinforcement learners may also have to deal with the problem of partial observability. That is, the current observation might not tell you everything about your current state. Say your cleaning robot found itself trapped in one of many identical closets in your house. Rescuing the robot involves inferring its precise location which might require considering earlier observations prior to it entering the closet.



MULTI-ARMED BANDIT

Exploit or explore

Finally, at any given point, reinforcement learners might know of one good policy, but there might be many other better policies that the agent has never tried. The reinforcement learner must constantly choose whether to exploit the best (currently) known strategy as a policy, or to explore the space of strategies, potentially giving up some short-term reward in exchange for knowledge.

The general reinforcement learning problem has a very general setting. Actions affect subsequent observations. Rewards are only observed when they correspond to the chosen actions. The environment may be either fully or partially observed. Accounting for all this complexity at once may be asking too much. Moreover, not every practical problem exhibits all this complexity. As a result, researchers have studied a number of special cases of reinforcement learning problems.

When the environment is fully observed, we call the reinforcement learning problem a Markov decision process. When the state does not depend on the previous actions, we call it a contextual bandit problem. When there is no state, just a set of available actions with initially unknown rewards, we have the classic multi-armed bandit problem.

Questions

1. Why is offline learning inappropriate for training an intelligent agent?
 - a. Because offline learning needs no data
 - b. Because the agent's actions affect its future observations

- c. Because agents cannot make predictions
 - d. Because offline learning is too slow
2. Spammers change their emails to get past spam filters. Which question about the environment does this illustrate?
- a. Does the environment remember what we did?
 - b. Does the environment want to help us?
 - c. Does the environment want to beat us?
 - d. Is the environment fully observed?
3. In chess, why must a reinforcement learner solve the credit assignment problem?
- a. The only real reward comes at the end, so it must work out which moves deserve credit or blame
 - b. The board is only partly visible
 - c. Every move receives its own reward
 - d. The rules change during the game
4. When there is no state, only a set of actions with initially unknown rewards, what is the problem called?
- a. A Markov decision process
 - b. A contextual bandit problem
 - c. Offline learning
 - d. A multi-armed bandit problem

Answers

1. **b)** Actions actually impact the environment, so a trained agent must account for how its actions affect its future observations, which offline learning ignores.

- 2. **c)** Spammers adapting their emails to evade spam filters is the passage's example of an environment that wants to beat us.
- 3. **a)** The only real reward, win or lose, comes at the end of the game, so the learner must determine which actions to credit or blame for the outcome.
- 4. **d)** With no state, just available actions with initially unknown rewards, we have the classic multi-armed bandit problem.

Practice

- 1. Explain in your own words what a policy is and what the goal of reinforcement learning is.

Hint: A policy connects what the agent sees to what it does.

Answer: A policy is a function that maps observations of the environment to actions; it governs how the agent behaves. The goal of reinforcement learning is to produce good policies, ones that earn high rewards over time.

- 2. Describe how a classification problem can be recast as reinforcement learning.

Hint: Think of each class as an action.

Answer: Create an agent with one action for each class, and an environment that gives a reward equal to the loss of the original supervised problem. Choosing the right class then amounts to choosing the action that earns the best reward.

- 3. A food-delivery app always recommends the restaurant that has worked best so far. Using the passage's terms, what is it doing, and what might it miss?

Hint: Think of exploit and explore.

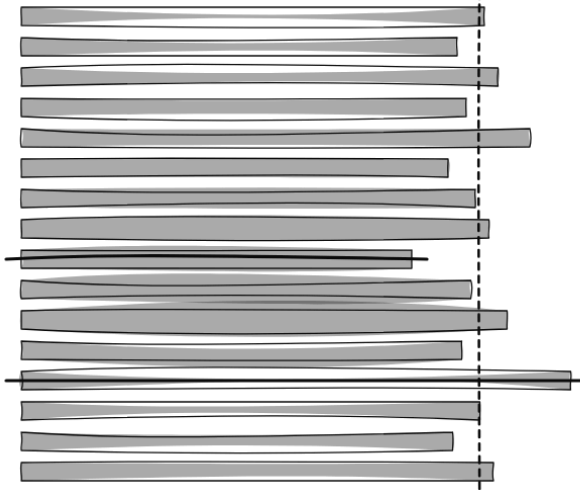
Answer: It is exploiting the best currently known policy. It might miss better restaurants it has never tried; exploring would give up some short-

term reward in exchange for knowledge about other options.

Roots

We have just reviewed a small subset of problems that machine learning can address. For a diverse set of machine learning problems, deep learning provides powerful tools for their solution. Although many deep learning methods are recent inventions, the core ideas behind learning from data have been studied for centuries. In fact, humans have held the desire to analyze data and to predict future outcomes for ages, and it is this desire that is at the root of much of natural science and mathematics. Two examples are the Bernoulli distribution, named after Jacob Bernoulli (1655 to 1705), and the Gaussian distribution discovered by Carl Friedrich Gauss (1777 to 1855). Gauss invented, for instance, the least mean squares algorithm, which is still used today for a multitude of problems from insurance calculations to medical diagnostics. Such tools enhanced the experimental approach in the natural sciences; for instance, Ohm's law relating current and voltage in a resistor is perfectly described by a linear model.

TRIMMED MEAN



Sixteen feet, one foot

Even in the middle ages, mathematicians had a keen intuition of estimates. For instance, the geometry book of Jacob Köbel (1460 to 1533) illustrates averaging the length of 16 adult men's feet to estimate the typical foot length in the population, as in the figure.

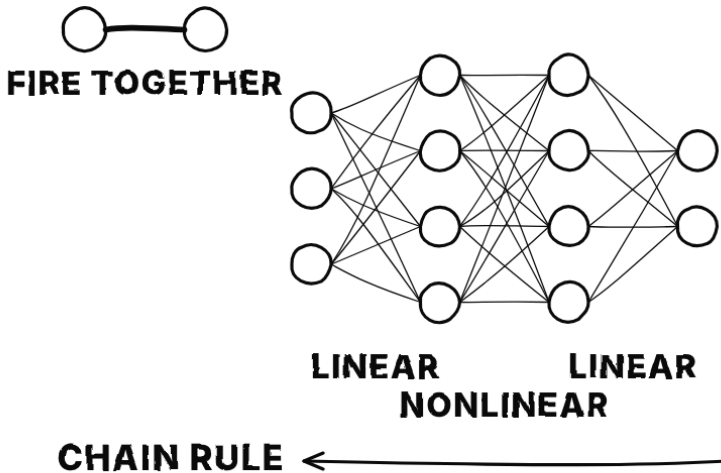
As a group of individuals exited a church, 16 adult men were asked to line up in a row and have their feet measured. The sum of these measurements was then divided by 16 to obtain an estimate for what now is called one foot. This "algorithm" was later improved to deal with misshapen feet: the two men with the shortest and longest feet were sent away, averaging only over the remainder. This is among the earliest examples of a trimmed mean estimate.

Statistics really took off with the availability and collection of data. One of its pioneers, Ronald Fisher (1890 to 1962), contributed significantly to its theory and also its applications in genetics. Many of his algorithms (such as linear discriminant analysis) and concepts (such as the Fisher information matrix) still hold a prominent place in the foundations of modern statistics. Even his data resources had a lasting impact. The Iris dataset that Fisher released in 1936 is still sometimes used to demonstrate machine learning algorithms. Fisher was also a proponent of eugenics, which should remind us that the morally dubious use of data science has as long and enduring a history as its productive use in industry and the natural sciences.

Other influences for machine learning came from the information theory of Claude Shannon (1916 to 2001) and the theory of computation proposed by Alan Turing (1912 to 1954). Turing posed the question "can machines think?" in his famous paper Computing Machinery and Intelligence. Describing what is now known as the Turing test, he proposed that a machine can be considered intelligent if it is difficult for a human evaluator to distinguish between the replies from a machine and those of a human, based purely on textual interactions.

Further influences came from neuroscience and psychology. After all, humans clearly exhibit intelligent behavior. Many scholars have asked whether one could explain and possibly reverse engineer this capacity. One of the first biologically inspired algorithms was formulated by Donald Hebb (1904 to 1985). In his groundbreaking book *The Organization of Behavior*, he posited that neurons learn by positive reinforcement. This became known as the Hebbian learning rule. These ideas inspired later work, such as Rosenblatt's perceptron learning algorithm, and laid the foundations of many stochastic

gradient descent algorithms that underpin deep learning today: reinforce desirable behavior and diminish undesirable behavior to obtain good settings of the parameters in a neural network.



Layers and backpropagation

Biological inspiration is what gave neural networks their name. For over a century (dating back to the models of Alexander Bain, 1873, and James Sherrington, 1890), researchers have tried to assemble computational circuits that resemble networks of interacting neurons. Over time, the interpretation of biology has become less literal, but the name stuck. At its heart lie a few key principles that can be found in most networks today. The first is the alternation of linear and nonlinear processing units, often referred to as layers. The second is

the use of the chain rule (also known as backpropagation) for adjusting parameters in the entire network at once.

After initial rapid progress, research in neural networks languished from around 1995 until 2005. This was mainly due to two reasons. First, training a network is computationally very expensive. While random-access memory was plentiful at the end of the past century, computational power was scarce. Second, datasets were relatively small. In fact, Fisher's Iris dataset from 1936 was still a popular tool for testing the efficacy of algorithms. The MNIST dataset with its 60,000 handwritten digits was considered huge.

Given the scarcity of data and computation, strong statistical tools such as kernel methods, decision trees, and graphical models proved empirically superior in many applications. Moreover, unlike neural networks, they did not require weeks to train and provided predictable results with strong theoretical guarantees.

Questions

1. Why did the improved foot-measuring procedure send away the men with the shortest and longest feet?
 - a. To make the line shorter
 - b. To deal with misshapen feet that would distort the average
 - c. Because only 14 men fitted in the row
 - d. To make the result a whole number
2. Which two principles, according to the passage, lie at the heart of most neural networks today?

- a. Trimmed means and least squares
 - b. Kernel methods and decision trees
 - c. Alternating linear and nonlinear layers, and backpropagation to adjust all parameters at once
 - d. The Turing test and Hebbian learning
3. Why did neural network research languish from around 1995 until 2005?
- a. Training was computationally expensive and datasets were small
 - b. Memory was too scarce to store networks
 - c. Neural networks had been proved not to work
 - d. Nobody knew about backpropagation
4. Why did kernel methods, decision trees and graphical models win in many applications at that time?
- a. They needed far more data than neural networks
 - b. They were biologically inspired
 - c. They were the only methods with layers
 - d. With little data and computation they worked better, trained fast, and had strong guarantees

Answers

1. **b)** The algorithm was improved to deal with misshapen feet: the two extremes were sent away and the average taken over the rest, an early trimmed mean.
2. **c)** The key principles are the alternation of linear and nonlinear processing units, called layers, and the chain rule, backpropagation, for adjusting parameters in the entire network at once.

- 3. a)** Two reasons: training was computationally very expensive while computing power was scarce, and datasets were relatively small.
- 4. d)** Given scarce data and computation they proved empirically superior, did not need weeks to train, and gave predictable results with strong theoretical guarantees.

Worked examples

Köbel's foot, plain and trimmed Sixteen men's feet measure, in centimetres: 27, 28, 29, 30, 30, 31, 31, 31, 32, 32, 33, 33, 34, 34, 35, 39. Find the plain average, then the trimmed mean after sending away the shortest and the longest.

Add all sixteen: the sum is 509 centimetres.

Divide by 16: $509 \div 16 \approx 31.81$ centimetres.

Send away the shortest (27) and the longest (39): the sum of the other fourteen is $509 - 27 - 39 = 443$.

Divide by 14: $443 \div 14 \approx 31.64$ centimetres; the unusual 39 no longer pulls the estimate up.

Answer: Plain average about 31.81 cm; trimmed mean about 31.64 cm

Practice

1. Five measurements are 10, 11, 12, 13 and 34. Compute the plain average and the trimmed mean that drops the smallest and largest. Which better describes a typical value?

Hint: Drop 10 and 34 for the trimmed mean.

Answer: Plain average: $80 \div 5 = 16$. Trimmed mean: $(11 + 12 + 13) \div 3 = 12$. The trimmed mean is more typical, because the one odd value 34 pulled the plain average up.

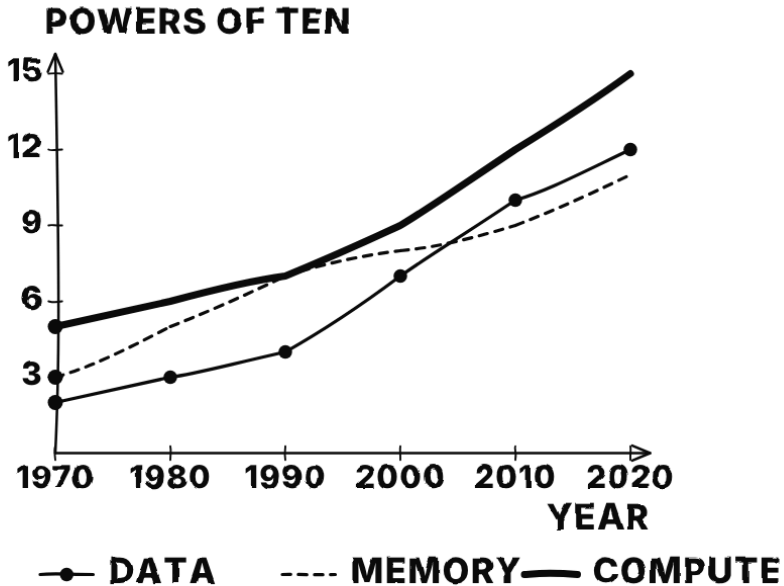
2. Describe the Turing test in your own words.

Hint: Who has to tell the difference, and from what?

Answer: A human evaluator exchanges text with a machine and with a human. If the evaluator finds it hard to tell the machine's replies from the human's, based only on the text, the machine can be considered intelligent.

The Road to Deep Learning

Much of this changed with the availability of massive amounts of data, thanks to the World Wide Web, the advent of companies serving hundreds of millions of users online, a dissemination of low-cost, high-quality sensors, inexpensive data storage (Kryder's law), and cheap computation (Moore's law). In particular, the landscape of computation in deep learning was revolutionized by advances in GPUs that were originally engineered for computer gaming. Suddenly algorithms and models that seemed computationally infeasible were within reach. This is best illustrated in the table, which sets dataset size against computer memory and computational power.



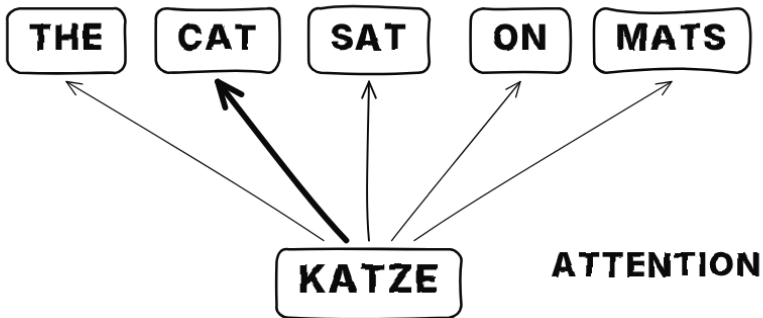
Data, memory and compute, 1970–2020

In 1970, a dataset held about 100 examples (the Iris data), memory was 1 KB, and a computer did about 100 K floating point calculations per second (the Intel 8080). In 1980, a dataset held about 1 K examples (house prices in Boston), memory was 100 KB, and computers did 1 MF calculations per second (the Intel 80186). In 1990, datasets held 10 K examples (optical character recognition), memory was 10 MB, and computers did 10 MF calculations per second (the Intel 80486). In 2000, datasets held 10 M examples (web pages), memory was 100 MB, and computers did 1 GF calculations per second (the Intel Core). In 2010, datasets held 10 G examples (advertising), memory was 1 GB, and computers did 1 TF calculations per second (the NVIDIA C2050). In 2020, datasets held 1 T examples

(a social network), memory was 100 GB, and computers did 1 PF calculations per second (the NVIDIA DGX-2).

Note that random-access memory has not kept pace with the growth in data. At the same time, increases in computational power have outpaced the growth in datasets. This means that statistical models need to become more memory efficient, and so they are free to spend more computer cycles optimizing parameters, thanks to the increased compute budget. Consequently, the sweet spot in machine learning and statistics moved from (generalized) linear models and kernel methods to deep neural networks. This is also one of the reasons why many of the mainstays of deep learning, such as multilayer perceptrons, convolutional neural networks, long short-term memory, and Q-Learning, were essentially "rediscovered" in the past decade, after lying comparatively dormant for considerable time.

The recent progress in statistical models, applications, and algorithms has sometimes been likened to the Cambrian explosion: a moment of rapid progress in the evolution of species. Indeed, the state of the art is not just a mere consequence of available resources applied to decades-old algorithms. Note that the list of ideas below barely scratches the surface of what has helped researchers achieve tremendous progress over the past decade.



TRANSFORMER: ATTENTION ONLY

Attention as a pointer

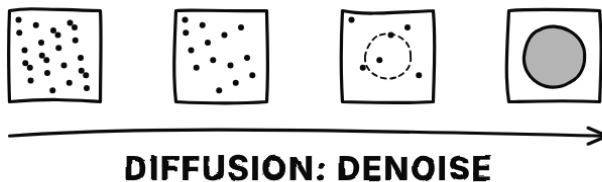
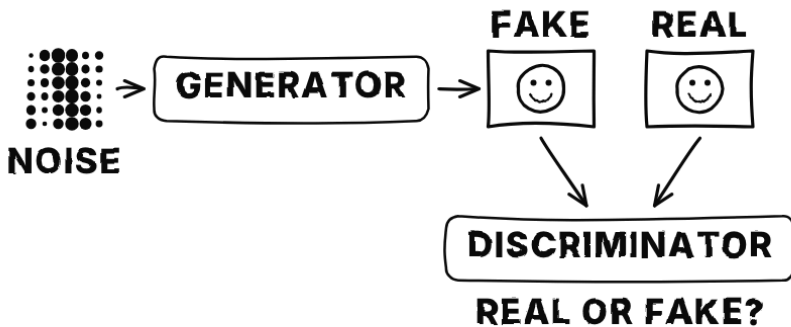
Novel methods for capacity control, such as dropout, have helped to mitigate overfitting. Here, noise is injected throughout the neural network during training.

Attention mechanisms solved a second problem that had plagued statistics for over a century: how to increase the memory and complexity of a system without increasing the number of learnable parameters. Researchers found an elegant solution by using what can only be viewed as a learnable pointer structure. Rather than having to remember an entire text sequence, for example for machine translation in a fixed-dimensional representation, all that needed to be stored was a pointer to the intermediate state of the translation process. This allowed for significantly increased accuracy for long

sequences, since the model no longer needed to remember the entire sequence before commencing the generation of a new one.

Built solely on attention mechanisms, the Transformer architecture has demonstrated superior scaling behavior: it performs better with an increase in dataset size, model size, and amount of training compute. This architecture has demonstrated compelling success in a wide range of areas, such as natural language processing, computer vision, speech recognition, reinforcement learning, and graph neural networks. For example, a single Transformer pretrained on modalities as diverse as text, images, joint torques, and button presses can play Atari, caption images, chat, and control a robot.

Modeling probabilities of text sequences, language models can predict text given other text. Scaling up the data, model, and compute has unlocked a growing number of capabilities of language models to perform desired tasks via human-like text generation based on input text. For instance, aligning language models with human intent, OpenAI's ChatGPT allows users to interact with it in a conversational way to solve problems, such as code debugging and creative writing.



Generator against discriminator

Multi-stage designs, for example via the memory networks and the neural programmer-interpreter, permitted statistical modelers to describe iterative approaches to reasoning. These tools allow for an internal state of the deep neural network to be modified repeatedly, thus carrying out subsequent steps in a chain of reasoning, just as a processor can modify memory for a computation.

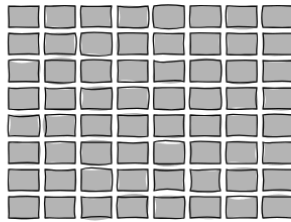
A key development in deep generative modeling was the invention of generative adversarial networks. Traditionally, statistical methods for density estimation and generative models focused on finding proper probability distributions and (often approximate) algorithms for sampling from them. As a result, these algorithms were largely limited by the lack of flexibility inherent in the statistical models. The crucial

innovation in generative adversarial networks was to replace the sampler by an arbitrary algorithm with differentiable parameters. These are then adjusted in such a way that the discriminator (effectively a two-sample test) cannot distinguish fake from real data. Through the ability to use arbitrary algorithms to generate data, density estimation was opened up to a wide variety of techniques. Examples of galloping zebras and of fake celebrity faces are each testimony to this progress. Even amateur doodlers can produce photorealistic images just based on sketches describing the layout of a scene.

Furthermore, while the diffusion process gradually adds random noise to data samples, diffusion models learn the denoising process to gradually construct data samples from random noise, reversing the diffusion process. They have started to replace generative adversarial networks in more recent deep generative models, such as in DALL-E 2 and Imagen for creative art and image generation based on text descriptions.

In many cases, a single GPU is insufficient for processing the large amounts of data available for training. Over the past decade the ability to build parallel and distributed training algorithms has improved significantly. One of the key challenges in designing scalable algorithms is that the workhorse of deep learning optimization, stochastic gradient descent, relies on relatively small minibatches of data to be processed. At the same time, small batches limit the efficiency of GPUs. Hence, training on 1,024 GPUs with a minibatch size of, say, 32 images per batch amounts to an aggregate minibatch of about 32,000 images. Work, first by Li (2017) and subsequently by You and colleagues (2017) and Jia and colleagues (2018), pushed the size up to 64,000 observations, reducing training time for the ResNet-

50 model on the ImageNet dataset to less than 7 minutes. By comparison, training times were initially of the order of days.



1,024 GPUS

$1,024 \times 32 \approx 32,000$
IMAGES PER STEP

DAYS



 **MINUTES**

THEANO → TENSORFLOW → PYTORCH, JAX

Many GPUs, one big batch

The ability to parallelize computation has also contributed to progress in reinforcement learning. This has led to significant progress in computers achieving superhuman performance on tasks like Go, Atari games, Starcraft, and in physics simulations (for example, using MuJoCo) where environment simulators are available. See, for example, Silver and colleagues (2016) for a description of such achievements in AlphaGo. In a nutshell, reinforcement learning works best if plenty of (state, action, reward) tuples are available. Simulation provides such an avenue.

Deep learning frameworks have played a crucial role in disseminating ideas. The first generation of open-source frameworks for neural network modeling consisted of Caffe, Torch, and Theano. Many seminal papers were written using these tools. These have now been superseded by TensorFlow (often used via its high-level API Keras), CNTK, Caffe 2, and Apache MXNet. The third generation of frameworks consists of so-called imperative tools for deep learning, a trend that was arguably ignited by Chainer, which used a syntax similar to Python NumPy to describe models. This idea was adopted by both PyTorch, the Gluon API of MXNet, and JAX.

Questions

1. Since memory grew more slowly than data but computation grew fast, what did statistical models have to do?
 - a. Use less computation and more memory
 - b. Become more memory efficient and spend more computer cycles optimizing parameters
 - c. Stop using large datasets
 - d. Return to linear models
2. What problem did attention mechanisms solve, according to the passage?
 - a. How to inject noise during training
 - b. How to train on many GPUs at once
 - c. How to increase a system's memory and complexity without increasing the number of learnable parameters

- d. How to sample from a probability distribution
-
- 3. In a generative adversarial network, how are the generator's parameters adjusted?
 - a. So that the discriminator cannot distinguish fake from real data
 - b. So that the discriminator always spots the fakes
 - c. So that it matches a fixed probability distribution
 - d. So that it adds noise step by step
-
- 4. Why does simulation help reinforcement learning?
 - a. It removes the need for rewards
 - b. It replaces the policy
 - c. It makes data independent
 - d. It supplies plenty of state, action, reward tuples

Answers

- 1. **b)** Statistical models need to become more memory efficient, and so they are free to spend more computer cycles optimizing parameters, thanks to the increased compute budget.
- 2. **c)** Attention solved how to increase memory and complexity without more learnable parameters, using a learnable pointer structure instead of remembering the whole sequence.
- 3. **a)** The sampler is replaced by an algorithm with differentiable parameters, adjusted so that the discriminator cannot distinguish fake from real data.
- 4. **d)** Reinforcement learning works best if plenty of state, action, reward tuples are available, and simulation provides such an avenue.

Worked examples

The aggregate minibatch Training runs on 1,024 GPUs, each processing a minibatch of 32 images. How many images make up the aggregate minibatch?

Each GPU processes 32 images per step.

There are 1,024 GPUs working at once.

$1,024 \times 32 = 32,768$ images, which the passage rounds to about 32,000.

Answer: 32,768 images, about 32,000

Practice

1. Using the table, by what factor did dataset size grow from 1970 (100 examples) to 2020 (1 T, a trillion examples), and by what factor did memory grow from 1 KB to 100 GB?

Hint: Count powers of ten; 1 GB is a million KB.

Answer: Data grew from 10^2 to 10^{12} , a factor of 10^{10} , ten billion.

Memory grew from 1 KB to 100 GB, that is 100 million KB, a factor of 10^8 .

So memory grew a hundred times less than data, which is why memory has not kept pace.

2. Explain in your own words how a diffusion model creates an image.

Hint: Start from what the diffusion process does to data.

Answer: The diffusion process gradually adds random noise to data until only noise is left. A diffusion model learns to undo this, removing noise step by step, so it can start from random noise and gradually construct a new data sample, such as an image matching a text description.

Success Stories

The division of labor between system researchers building better tools and statistical modelers building better neural networks has greatly simplified things. For instance, training a linear logistic regression model used to be a nontrivial homework problem, worthy to give to new machine learning Ph.D. students at Carnegie Mellon University in 2014. By now, this task can be accomplished with under 10 lines of code, putting it firmly within the reach of any programmer.

Artificial intelligence has a long history of delivering results that would be difficult to accomplish otherwise. For instance, mail sorting systems using optical character recognition have been deployed since the 1990s. This is, after all, the source of the famous MNIST dataset of handwritten digits. The same applies to reading checks for bank deposits and scoring creditworthiness of applicants. Financial transactions are checked for fraud automatically. This forms the backbone of many e-commerce payment systems, such as PayPal, Stripe, AliPay, WeChat, Apple, Visa, and MasterCard. Computer programs for chess have been competitive for decades. Machine learning feeds search, recommendation, personalization, and ranking on the Internet. In other words, machine learning is pervasive, albeit often hidden from sight.

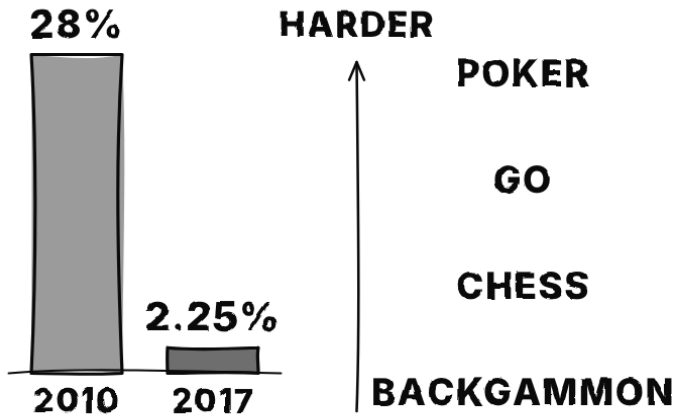
It is only recently that AI has been in the limelight, mostly due to solutions to problems that were considered intractable previously and that are directly related to consumers. Many of such advances are attributed to deep learning.

Intelligent assistants, such as Apple's Siri, Amazon's Alexa, and Google's assistant, are able to respond to spoken requests with a reasonable degree of accuracy. This includes menial jobs, like turning on light switches, and more complex tasks, such as arranging barber's appointments and offering phone support dialog. This is likely the most noticeable sign that AI is affecting our lives.

A key ingredient in digital assistants is their ability to recognize speech accurately. The accuracy of such systems has gradually increased to the point of achieving parity with humans for certain applications.

Object recognition has likewise come a long way. Identifying the object in a picture was a fairly challenging task in 2010. On the ImageNet benchmark researchers from NEC Labs and University of Illinois at Urbana-Champaign achieved a top-five error rate of 28%. By 2017, this error rate was reduced to 2.25%. Similarly, stunning results have been achieved for identifying birdsong and for diagnosing skin cancer.

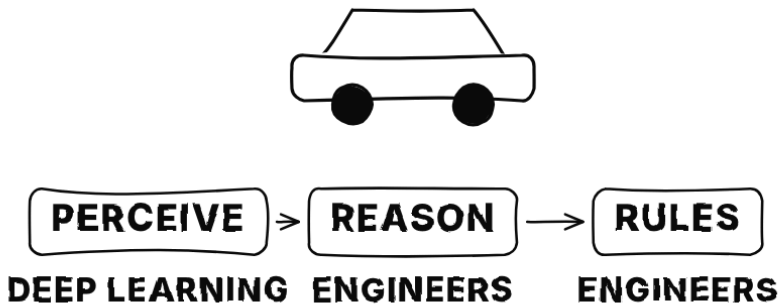
IMAGENET ERROR



From 28% to 2.25%, and games

Prowess in games used to provide a measuring stick for human ability. Starting from TD-Gammon, a program for playing backgammon using temporal difference reinforcement learning, algorithmic and computational progress has led to algorithms for a wide range of applications. Compared with backgammon, chess has a much more complex state space and set of actions. DeepBlue beat Garry Kasparov using massive parallelism, special-purpose hardware and efficient search through the game tree. Go is more difficult still, due to its huge state space. AlphaGo reached human parity in 2015, using deep learning combined with Monte Carlo tree sampling. The challenge in Poker was that the state space is large and only partially observed (we do not know the opponents' cards). Libratus exceeded human performance in Poker using efficiently structured strategies.

Another indication of progress in AI is the advent of self-driving vehicles. While full autonomy is not yet within reach, excellent progress has been made in this direction, with companies such as Tesla, NVIDIA, and Waymo shipping products that enable partial autonomy. What makes full autonomy so challenging is that proper driving requires the ability to perceive, to reason and to incorporate rules into a system. At present, deep learning is used primarily in the visual aspect of these problems. The rest is heavily tuned by engineers.



BUILT FOR ONE GOAL

Driving needs more than vision

This barely scratches the surface of significant applications of machine learning. For instance, robotics, logistics, computational biology, particle physics, and astronomy owe some of their most

impressive recent advances at least in parts to machine learning, which is thus becoming a ubiquitous tool for engineers and scientists.

Frequently, questions about a coming AI apocalypse and the plausibility of a singularity have been raised in non-technical articles. The fear is that somehow machine learning systems will become sentient and make decisions, independently of their programmers, that directly impact the lives of humans. To some extent, AI already affects the livelihood of humans in direct ways: creditworthiness is assessed automatically, autopilots mostly navigate vehicles, decisions about whether to grant bail use statistical data as input. More frivolously, we can ask Alexa to switch on the coffee machine.

Fortunately, we are far from a sentient AI system that could deliberately manipulate its human creators. First, AI systems are engineered, trained, and deployed in a specific, goal-oriented manner. While their behavior might give the illusion of general intelligence, it is a combination of rules, heuristics and statistical models that underlie the design. Second, at present, there are simply no tools for artificial general intelligence that are able to improve themselves, reason about themselves, and that are able to modify, extend, and improve their own architecture while trying to solve general tasks.

A much more pressing concern is how AI is being used in our daily lives. It is likely that many routine tasks, currently fulfilled by humans, can and will be automated. Farm robots will likely reduce the costs for organic farmers but they will also automate harvesting operations. This phase of the industrial revolution may have profound consequences for large swaths of society, since menial jobs provide much employment in many countries. Furthermore, statistical models, when applied without care, can lead to racial, gender, or age

bias and raise reasonable concerns about procedural fairness if automated to drive consequential decisions. It is important to ensure that these algorithms are used with care. With what we know today, this strikes us as a much more pressing concern than the potential of malevolent superintelligence for destroying humanity.

Questions

1. Why does the passage call machine learning "pervasive, albeit often hidden from sight"?
 - a. Because it is used only in research laboratories
 - b. Because it quietly runs mail sorting, fraud checks, search and recommendations
 - c. Because people refuse to use it
 - d. Because it only became possible in 2017
2. What made Poker hard for computers, according to the passage?
 - a. Its state space is tiny
 - b. It requires special-purpose hardware
 - c. Its state space is large and only partially observed, since opponents' cards are hidden
 - d. It has no rewards
3. What is the main reason the authors give for not fearing a sentient AI that manipulates its creators?
 - a. AI systems are engineered for specific goals, and no tools yet exist for self-improving general intelligence

- b. AI systems are always switched off at night
 - c. AI has never been used in real decisions
 - d. Laws forbid AI from making decisions
4. What do the authors consider a more pressing concern than a malevolent superintelligence?
- a. That computers will run out of memory
 - b. That games will become too easy
 - c. That AI will stop improving
 - d. Automation of routine jobs and biased models making consequential decisions

Answers

1. **b)** Mail sorting, check reading, credit scoring, fraud checks, search, recommendation and ranking all use machine learning, mostly unseen.
2. **c)** The challenge in Poker was that the state space is large and only partially observed: we do not know the opponents' cards.
3. **a)** AI systems are engineered and deployed in a specific, goal-oriented manner, and there are no tools for artificial general intelligence that can improve and modify themselves.
4. **d)** The pressing concern is how AI is used in daily life: automation of many routine tasks, and models applied without care leading to racial, gender or age bias.

Worked examples

Progress on ImageNet On ImageNet the top-five error rate was 28% in 2010 and 2.25% by 2017. Out of 1,000 test pictures, about how

many were missed in each year, and how many times smaller did the error become?

In 2010: 28% of 1,000 is $0.28 \times 1,000 = 280$ pictures missed.

By 2017: 2.25% of 1,000 is $0.0225 \times 1,000 = 22.5$, about 22 or 23 pictures missed.

Ratio: $28 \div 2.25 \approx 12.4$.

Answer: About 280 missed in 2010 and about 22 in 2017; the error became roughly 12 times smaller

Practice

1. Explain why full self-driving is harder than recognizing objects in pictures, according to the passage.

Hint: Driving needs more than seeing.

Answer: Proper driving requires the ability to perceive, to reason and to incorporate rules into a system. Deep learning is mainly used for the visual part, perceiving; the rest is still heavily tuned by engineers.

2. Give three ways, named in the passage, in which AI already affects people's lives directly.

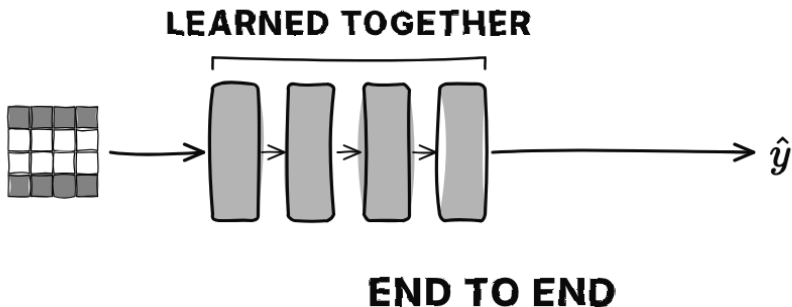
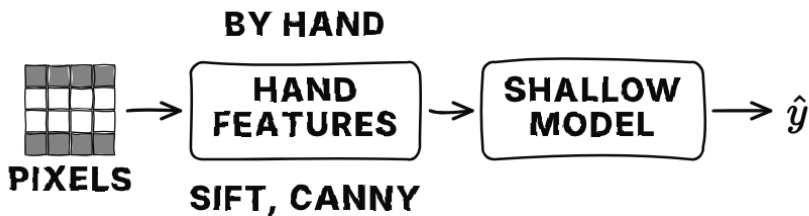
Hint: Think of money, travel and the courts.

Answer: Creditworthiness is assessed automatically, autopilots mostly navigate vehicles, and decisions about whether to grant bail use statistical data as input.

The Essence of Deep Learning

Thus far, we have talked in broad terms about machine learning. Deep learning is the subset of machine learning concerned with models based on many-layered neural networks. It is deep in precisely the sense that its models learn many layers of transformations. While this might sound narrow, deep learning has given rise to a dizzying array of models, techniques, problem formulations, and applications. Many intuitions have been developed to explain the benefits of depth.

Arguably, all machine learning has many layers of computation, the first consisting of feature processing steps. What differentiates deep learning is that the operations learned at each of the many layers of representations are learned jointly from data.



Hand-made features versus end to end

The problems that we have discussed so far, such as learning from the raw audio signal, the raw pixel values of images, or mapping between sentences of arbitrary lengths and their counterparts in foreign languages, are those where deep learning excels and traditional methods falter. It turns out that these many-layered models are capable of addressing low-level perceptual data in a way that previous tools could not. Arguably the most significant commonality in deep learning methods is end-to-end training. That is, rather than assembling a system based on components that are individually tuned, one builds the system and then tunes their performance jointly. For instance, in computer vision scientists used to separate the process of feature engineering from the process of building machine learning models. The Canny edge detector and Lowe's SIFT

feature extractor reigned supreme for over a decade as algorithms for mapping images into feature vectors. In bygone days, the crucial part of applying machine learning to these problems consisted of coming up with manually-engineered ways of transforming the data into some form amenable to shallow models. Unfortunately, there is only so much that humans can accomplish by ingenuity in comparison with a consistent evaluation over millions of choices carried out automatically by an algorithm. When deep learning took over, these feature extractors were replaced by automatically tuned filters that yielded superior accuracy.

Thus, one key advantage of deep learning is that it replaces not only the shallow models at the end of traditional learning pipelines, but also the labor-intensive process of feature engineering. Moreover, by replacing much of the domain-specific preprocessing, deep learning has eliminated many of the boundaries that previously separated computer vision, speech recognition, natural language processing, medical informatics, and other application areas, thereby offering a unified set of tools for tackling diverse problems.

Beyond end-to-end training, we are experiencing a transition from parametric statistical descriptions to fully nonparametric models. When data is scarce, one needs to rely on simplifying assumptions about reality in order to obtain useful models. When data is abundant, these can be replaced by nonparametric models that better fit the data. To some extent, this mirrors the progress that physics experienced in the middle of the previous century with the availability of computers. Rather than solving by hand parametric approximations of how electrons behave, one can now resort to numerical simulations of the associated partial differential equations.

This has led to much more accurate models, albeit often at the expense of interpretation.

Another difference from previous work is the acceptance of suboptimal solutions, dealing with nonconvex nonlinear optimization problems, and the willingness to try things before proving them. This new-found empiricism in dealing with statistical problems, combined with a rapid influx of talent has led to rapid progress in the development of practical algorithms, albeit in many cases at the expense of modifying and re-inventing tools that existed for decades.

In the end, the deep learning community prides itself on sharing tools across academic and corporate boundaries, releasing many excellent libraries, statistical models, and trained networks as open source. It is in this spirit that the notebooks forming this book are freely available for distribution and use. We have worked hard to lower the barriers of access for anyone wishing to learn about deep learning and we hope that our readers will benefit from this.

In summary: machine learning studies how computer systems can leverage experience (often data) to improve performance at specific tasks. It combines ideas from statistics, data mining, and optimization. Often, it is used as a means of implementing AI solutions. As a class of machine learning, representational learning focuses on how to automatically find the appropriate way to represent data. Considered as multi-level representation learning through learning many layers of transformations, deep learning replaces not only the shallow models at the end of traditional machine learning pipelines, but also the labor-intensive process of feature engineering. Much of the recent progress in deep learning has been triggered by an abundance of data arising from cheap sensors and Internet-scale

applications, and by significant progress in computation, mostly through GPUs. Furthermore, the availability of efficient deep learning frameworks has made design and implementation of whole system optimization significantly easier, and this is a key component in obtaining high performance.

Questions

1. Since all machine learning arguably has many layers of computation, what sets deep learning apart?
 - a. It uses only one layer
 - b. The operations at each of its many layers of representations are learned jointly from data
 - c. It never uses features
 - d. It always needs a human to design each layer
2. What does end-to-end training mean?
 - a. Building the whole system and tuning its performance jointly
 - b. Tuning each component separately and then assembling them
 - c. Training only the last layer of a model
 - d. Training on a dataset from start to end once
3. Why did automatically tuned filters beat hand-engineered feature extractors such as SIFT?
 - a. Humans stopped working on vision
 - b. Hand-engineered features were too fast

- c. An algorithm can consistently evaluate millions of choices, more than human ingenuity can
 - d. Feature extractors were illegal to use
4. When can simplifying assumptions about reality be replaced by nonparametric models?
- a. When data is scarce
 - b. When models must be easy to interpret
 - c. When there are no computers
 - d. When data is abundant

Answers

1. **b)** What differentiates deep learning is that the operations learned at each of the many layers of representations are learned jointly from data.
2. **a)** Rather than assembling a system from individually tuned components, one builds the system and then tunes its performance jointly.
3. **c)** There is only so much humans can accomplish by ingenuity compared with a consistent evaluation over millions of choices carried out automatically by an algorithm.
4. **d)** When data is scarce one must rely on simplifying assumptions; when data is abundant these can be replaced by nonparametric models that fit the data better.

Practice

1. Explain in your own words the two things deep learning replaces in a traditional machine learning pipeline.

Hint: Think of what happens before and at the end of the pipeline.

Answer: It replaces the shallow model at the end of the pipeline, and also the labor-intensive feature engineering at the start, where people designed by hand how to turn raw data, such as images, into feature vectors.

2. The passage compares the shift to nonparametric models with a change in physics. Describe the comparison and the price paid in both cases.

Hint: What did computers let physicists do instead of solving by hand?

Answer: Physicists stopped solving simplified parametric approximations of how electrons behave by hand and turned to numerical simulations of the underlying equations, just as machine learning moves from simplified parametric models to models that fit abundant data. Both give much more accurate models, often at the expense of interpretation.