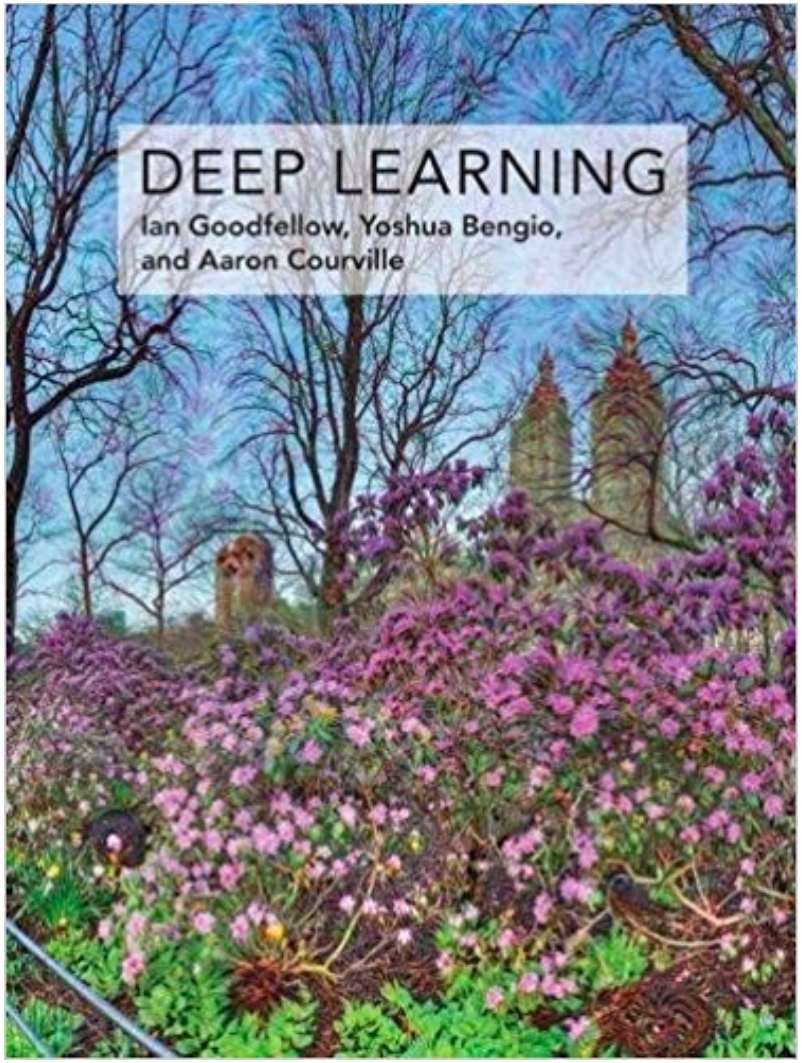
國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Deep Learning
深度學習
Fall 2023

Machine Learning Basics
(Chapter 5.1-5.3)

Prof. Chia-Han Lee
李佳翰 教授



- Figure source: Textbook and Internet
- You are encouraged to buy the textbook.
- Please respect the copyright of the textbook. Do not distribute the materials to other people.

Learning algorithms

- A machine learning algorithm is an algorithm that is able to learn from data.
- What do we mean by learning?
- Mitchell in 1997 provides a succinct definition: “A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*.”

The task, *T*

- The process of learning itself is not the task. Learning is our means of attaining the ability to perform the task. For example, if we want a robot to be able to walk, then walking is the task.

- Machine learning tasks are usually described in terms of how the machine learning system should process an example. An example is a collection of features that have been quantitatively measured from some object or event.

- We typically represent an example as a vector $\boldsymbol{x} \in \mathbb{R}^n$ where each entry x_i of the vector is another feature. For example, the features of an image are usually the values of the pixels in the image.

The task, *T*

- Classification: The computer program is asked to specify which of k categories some input belongs to.

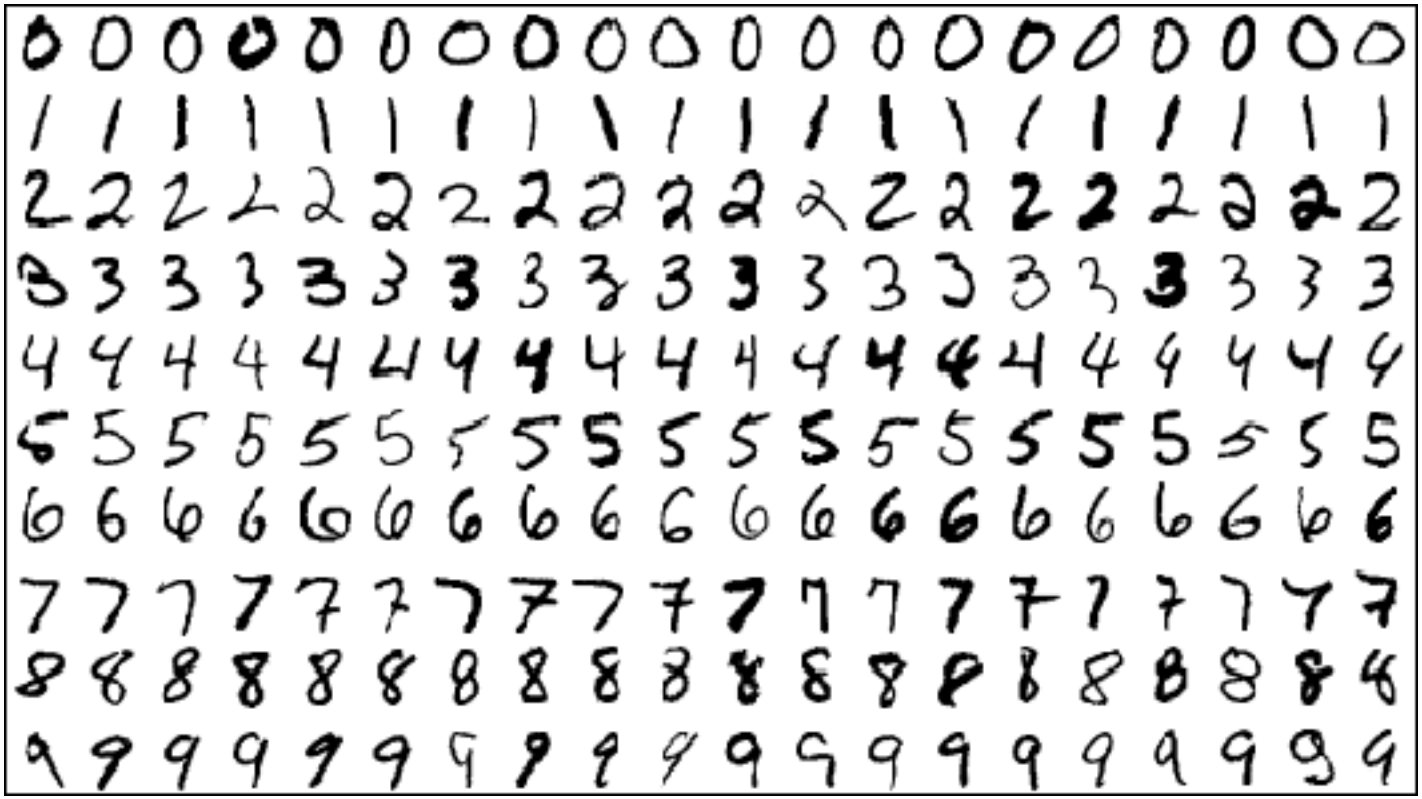
- To solve this task, the learning algorithm is usually asked to produce a function $f : \mathbb{R}^n \rightarrow \{1, \ldots, k\}$. When $y = f(\boldsymbol{x})$, the model assigns an input described by vector $\boldsymbol{x}$ to a category identified by numeric code y.

- There are other variants of the classification task, for example, where f outputs a probability distribution over classes.

- An example of a classification task is object recognition, where the input is an image (usually described as a set of pixel brightness values), and the output is a numeric code identifying the object in the image.

The task, *T*

- **Handwriting detection**



http://myselph.de/mnistExamples.png

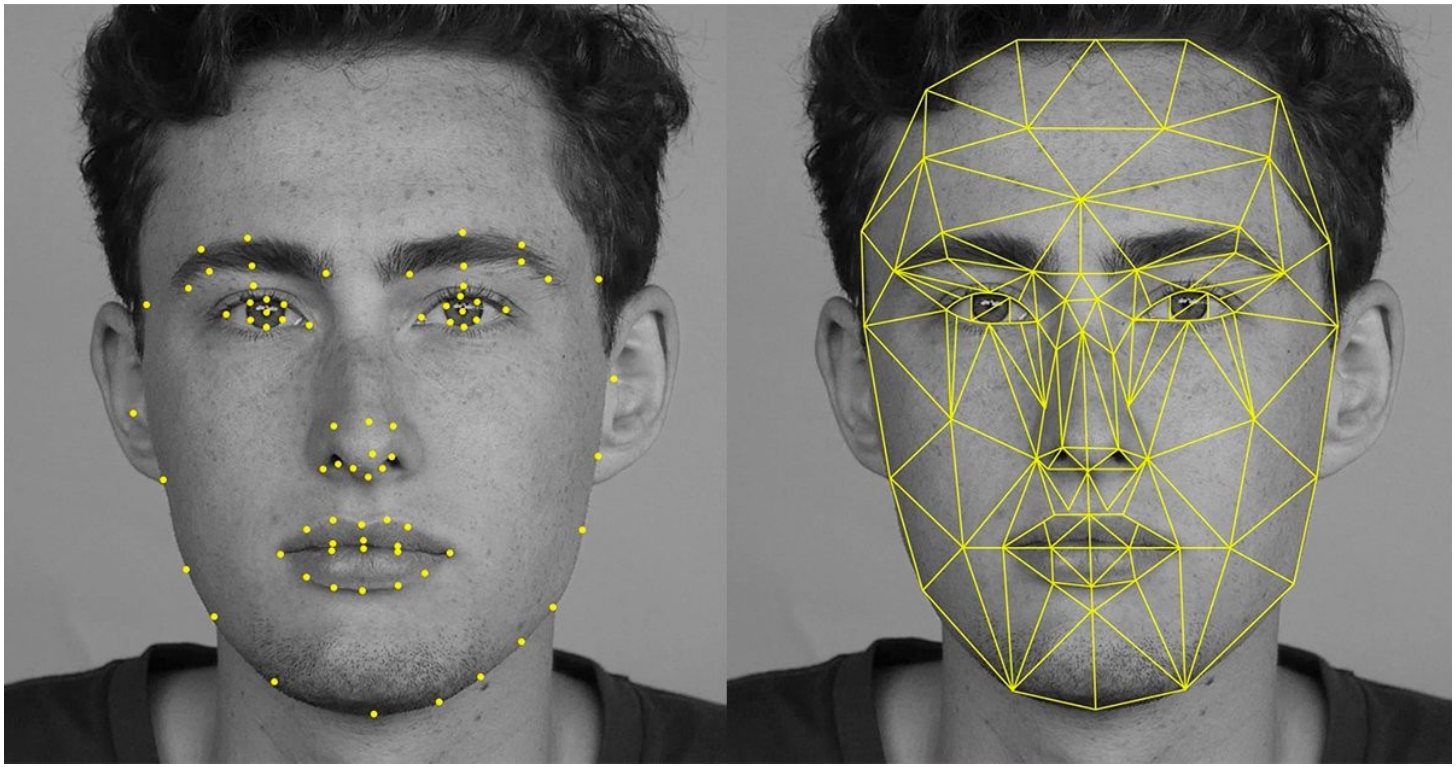
The task, *T*

- **Fingerprint detection**

http://biometrics.mainguet.org/types/fingerprint/algo/whorl_loop_arche.jpg
http://biometrics.mainguet.org/types/fingerprint/algo/minutiae.jpg

The task, *T*

- **Face recognition**



https://cdn-images-1.medium.com/max/1600/1*wh1N-kogDMaZYS17lqyqeQ.jpeg

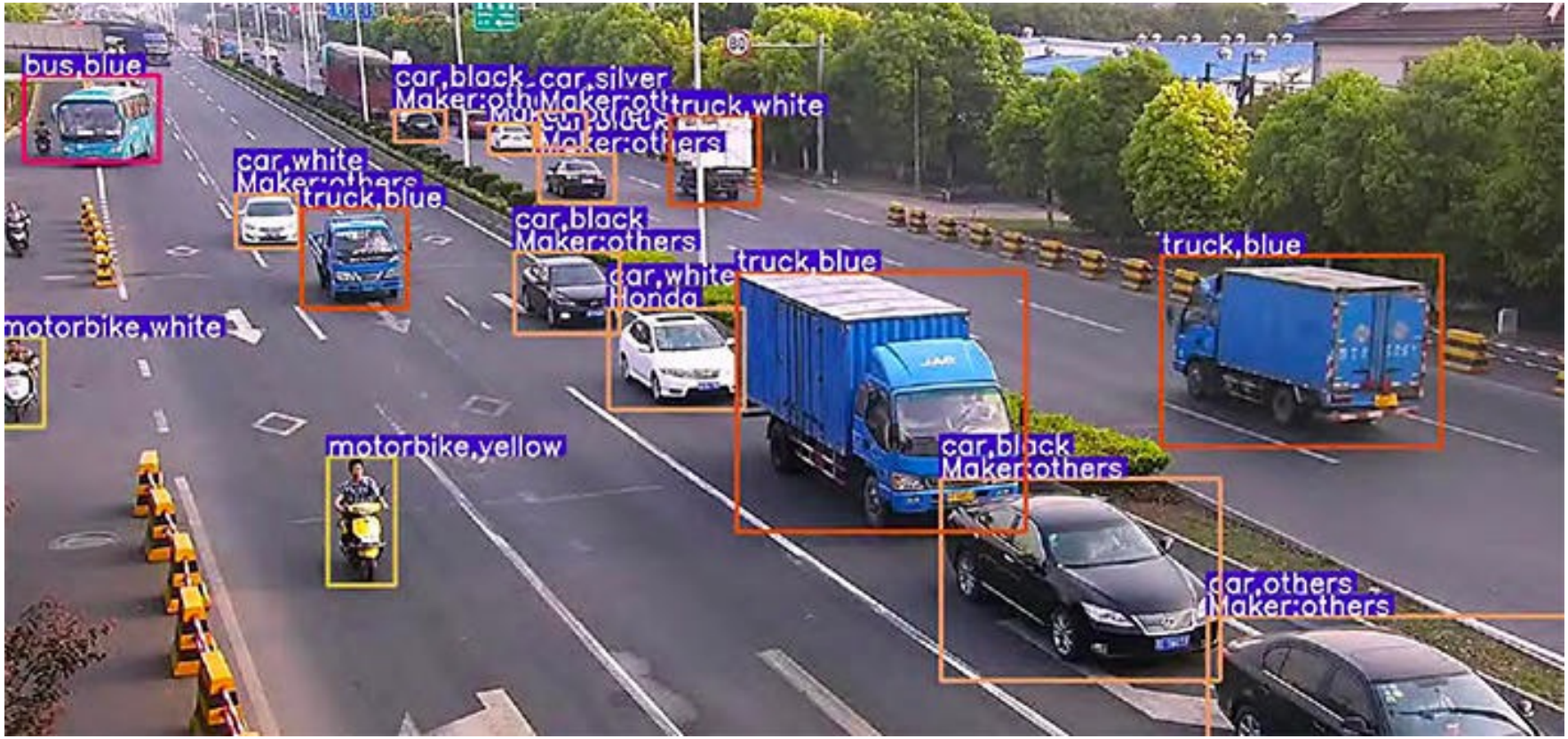
The task, *T*

- **Object recognition**

https://d26pej6xfl28et.cloudfront.net/wp-content/uploads/2017/08/AI_Object_Recognition_Feature_Img.jpg?x89106

The task, *T*

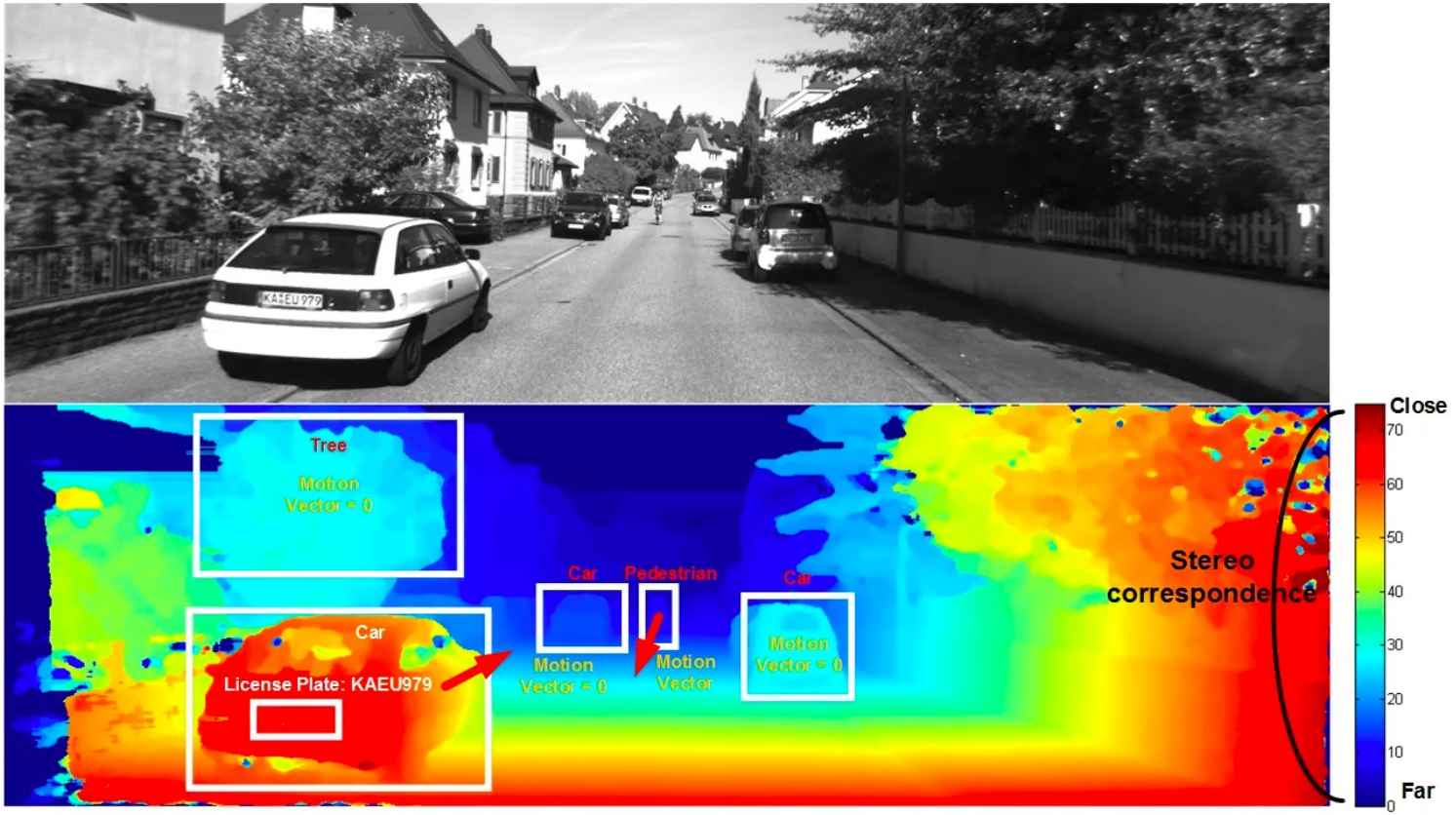
- **Surveillance camera**



http://journal.jp.fujitsu.com/en/2016/10/17/01/img/20161017_01_index_pic_10.jpg

The task, *T*

- **Road detection, pedestrian detection for self-driving cars**



https://i2.wp.com/kim.engin.umich.edu/wp-content/uploads/sites/355/2017/12/computervision.png

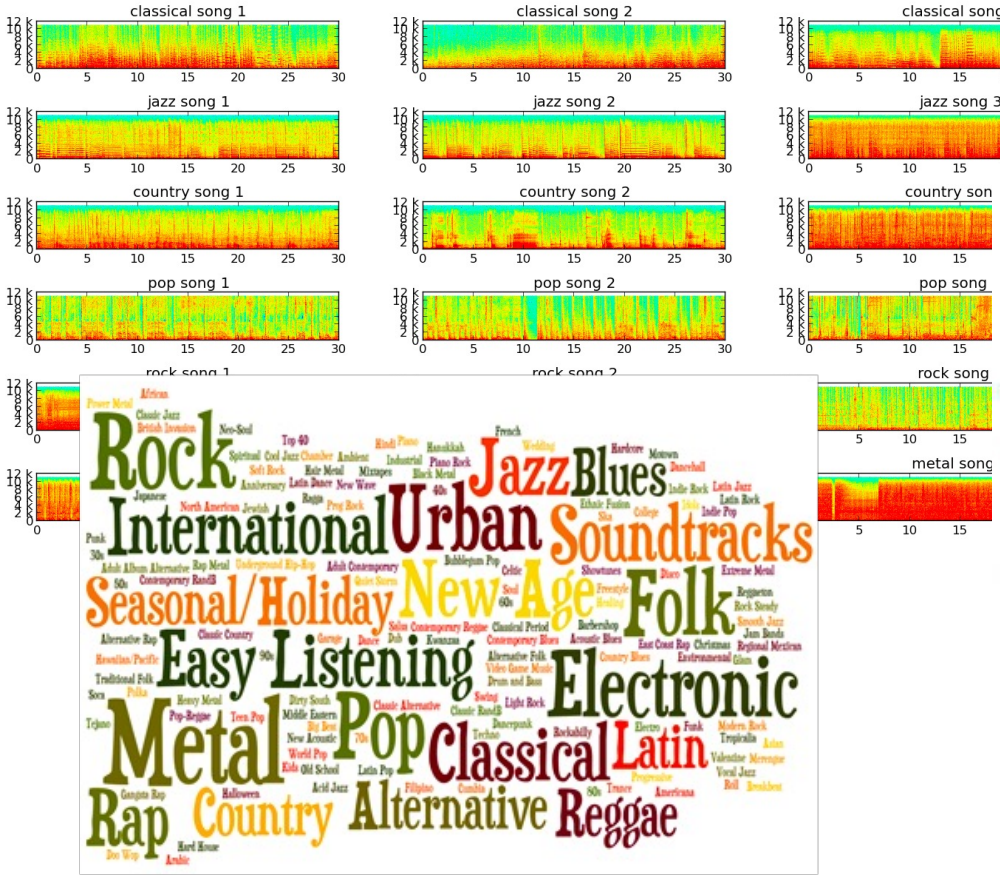
The task, *T*

- **Spam email filter**

http://arkansasconsumer.org/wp-content/uploads/2016/04/Spam.jpg

The task, *T*

- **Music genre classification**



The confusion matrix was plotted for this in Fig. 6.

	blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock
blues	61.00% (61)	2.00% (2)	12.00% (12)	0	2.00% (2)	3.00% (3)	8.00% (8)	0	2.00% (2)	10.00% (10)
classical	0	92.00% (92)	0	0	1.00% (1)	6.00% (6)	0	0	0	1.00% (1)
country	5.00% (5)	0	61.00% (61)	9.00% (9)	2.00% (2)	2.00% (2)	1.00% (1)	3.00% (3)	6.00% (6)	11.00% (11)
disco	1.00% (1)	0	5.00% (5)	56.00% (56)	3.00% (3)	2.00% (2)	4.00% (4)	8.00% (8)	8.00% (8)	13.00% (13)
hiphop	2.00% (2)	3.00% (3)	4.00% (4)	5.00% (5)	58.00% (58)	0	7.00% (7)	5.00% (5)	13.00% (13)	3.00% (3)
jazz	3.00% (3)	9.00% (9)	7.00% (7)	1.00% (1)	0	72.00% (72)	1.00% (1)	0	2.00% (2)	5.00% (5)
metal	3.00% (3)	0	0	2.00% (2)	4.00% (4)	2.00% (2)	80.00% (80)	0	1.00% (1)	8.00% (8)
pop	0	2.00% (2)	10.00% (10)	9.00% (9)	6.00% (6)	0	0	70.00% (70)	3.00% (3)	0
reggae	5.00% (5)	3.00% (3)	10.00% (10)	7.00% (7)	12.00% (12)	1.00% (1)	1.00% (1)	3.00% (3)	55.00% (55)	3.00% (3)
rock	9.00% (9)	1.00% (1)	10.00% (10)	19.00% (19)	1.00% (1)	6.00% (6)	11.00% (11)	1.00% (1)	5.00% (5)	37.00% (37)

Fig. 7. SVM with reduced dimensions.

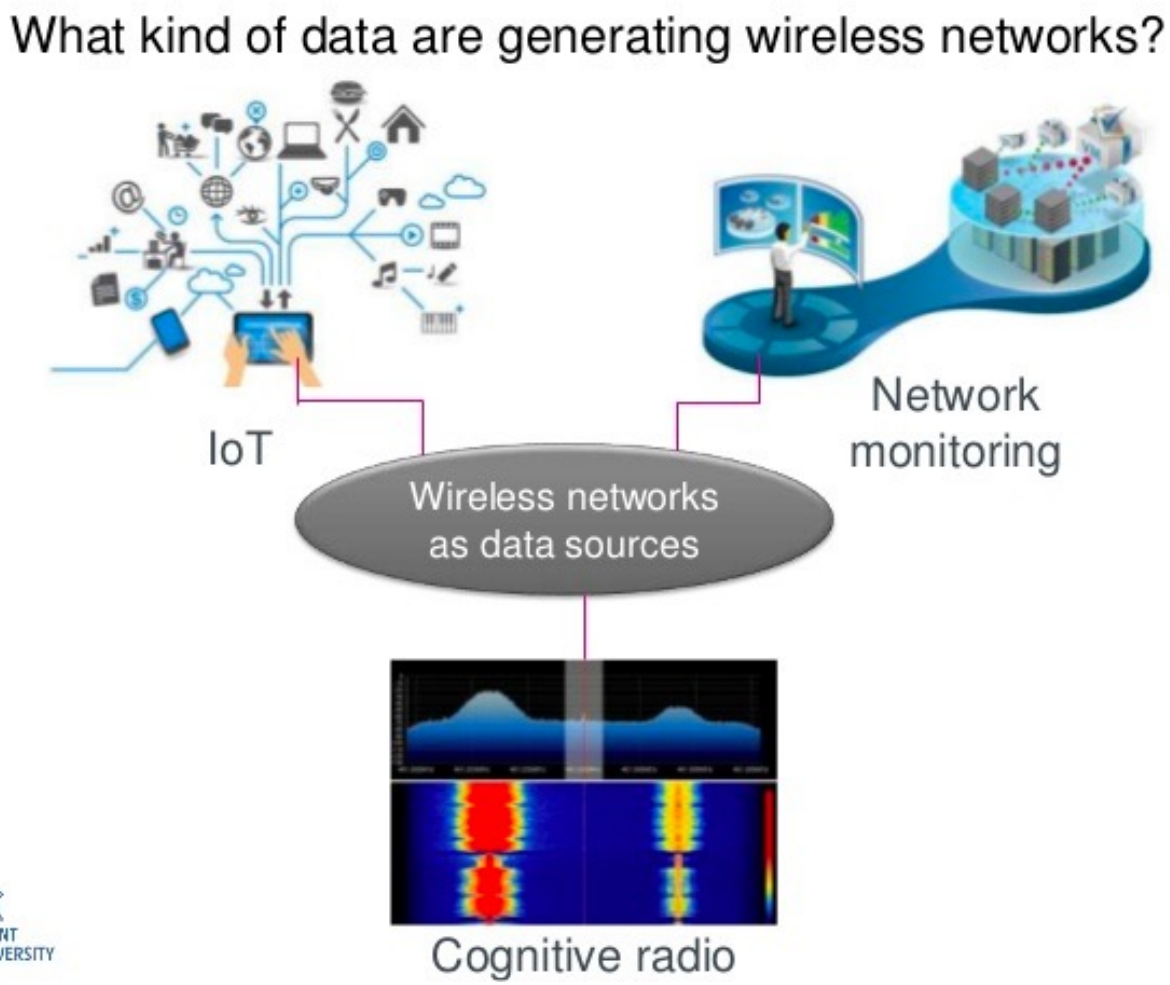
https://raw.githubusercontent.com/jazdev/genreXpose/master/genreXpose/graphs/Spectrogram_Genres_clean.png
https://ai2-s2-public.s3.amazonaws.com/figures/2017-08-08/e272ce9dd8c7adec73817135d053f5819d968082/6-Figure7-1.png
http://www.southernmuseumofmusic.com/Spotlight/01-Genre/0-images/History-01.png

The task, *T*

- **Recommendation system: Music/movie/video/shopping recommendation**

https://o.aolcdn.com/images/dims?quality=100&image_uri=http%3A%2F%2Fwww.blogcdn.com%2Fwww.engadget.com%2Fmedia%2F2012%2F02%2Fappletv2.020312.jpg&client=cbc79c14efcebee57402&signature=8eeeeec4130169746e4a35a1cf0793fce75f196f

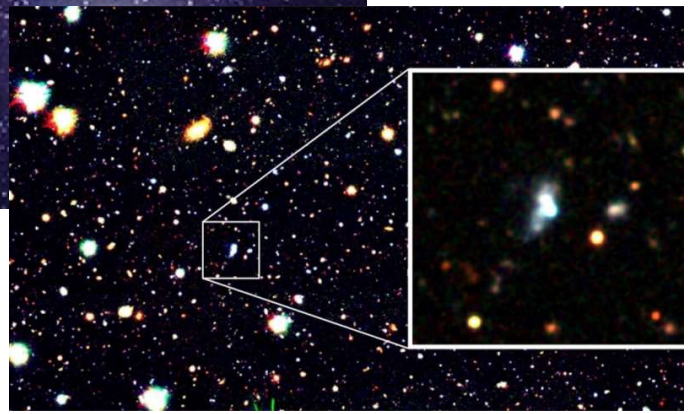
The task, *T*



https://image.slidesharecdn.com/ttjg6jg4qdcl9sawvdvg-signature-78f69696f964c9cc253b4acd4ecb196acc4a51da7692bbe4ca18825f7aca38c4-poli-161128140921/95/machine-learning-for-wireless-networks-bestcom2016-5-638.jpg?cb=1485252301

The task, *T*

- **Astronomy**



https://www.illinoistimes.com/springfield/astronomy/Content?oid=11493849
https://phys.org/news/2020-08-machine-early-galaxy.html

The task, *T*

https://www.freecodecamp.org/news/chihuahua-or-muffin-my-search-for-the-best-computer-vision-api-cbda4d6b425d/

The task, *T*

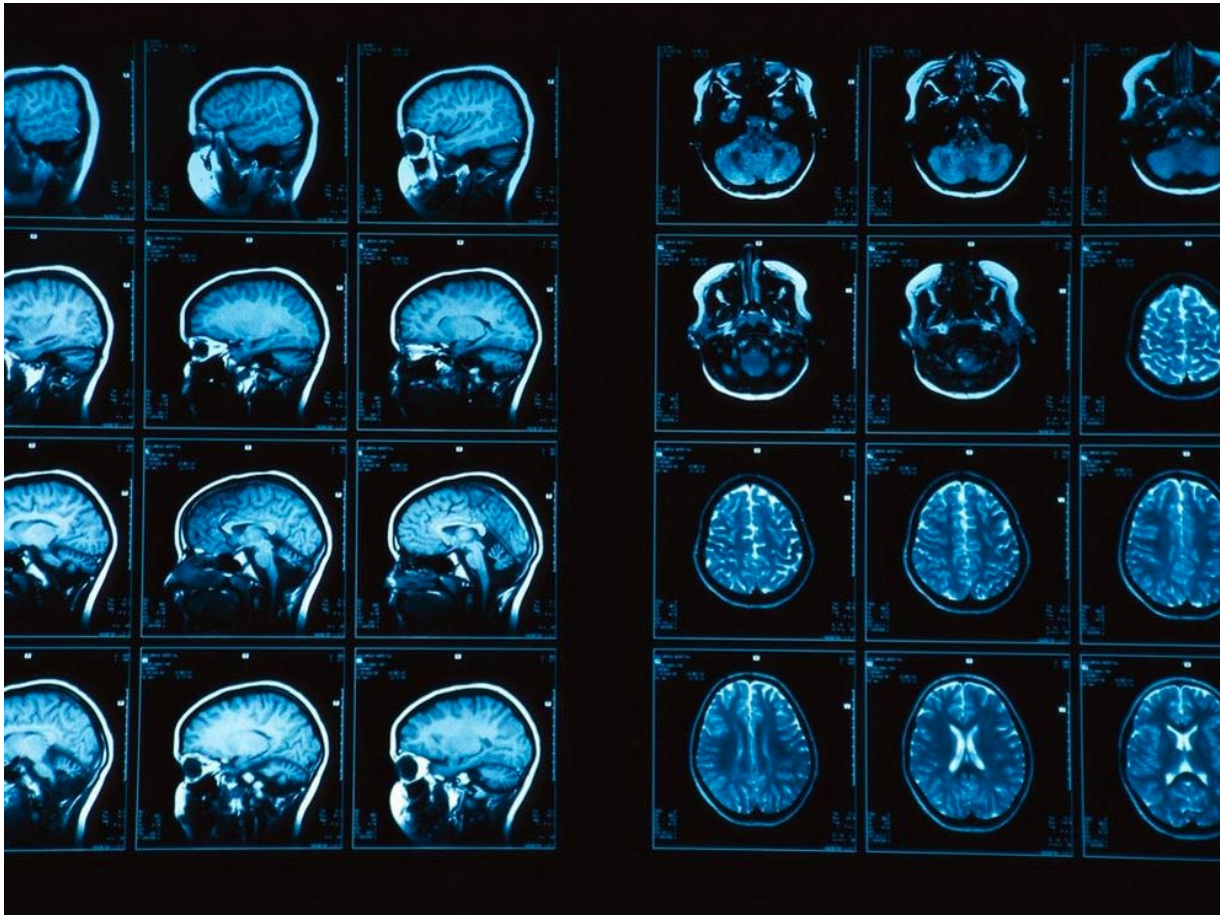
- Classification with missing inputs: When some of the inputs may be missing, rather than providing a single classification function, the learning algorithm must learn a set of functions. Each function corresponds to classifying $\boldsymbol{x}$ with a different subset of its inputs missing.

- This kind of situation arises frequently in medical diagnosis, because many kinds of medical tests are expensive or invasive.

- One way to efficiently define such a large set of functions is to learn a probability distribution over all the relevant variables, then solve the classification task by marginalizing out the missing variables.

The task, *T*

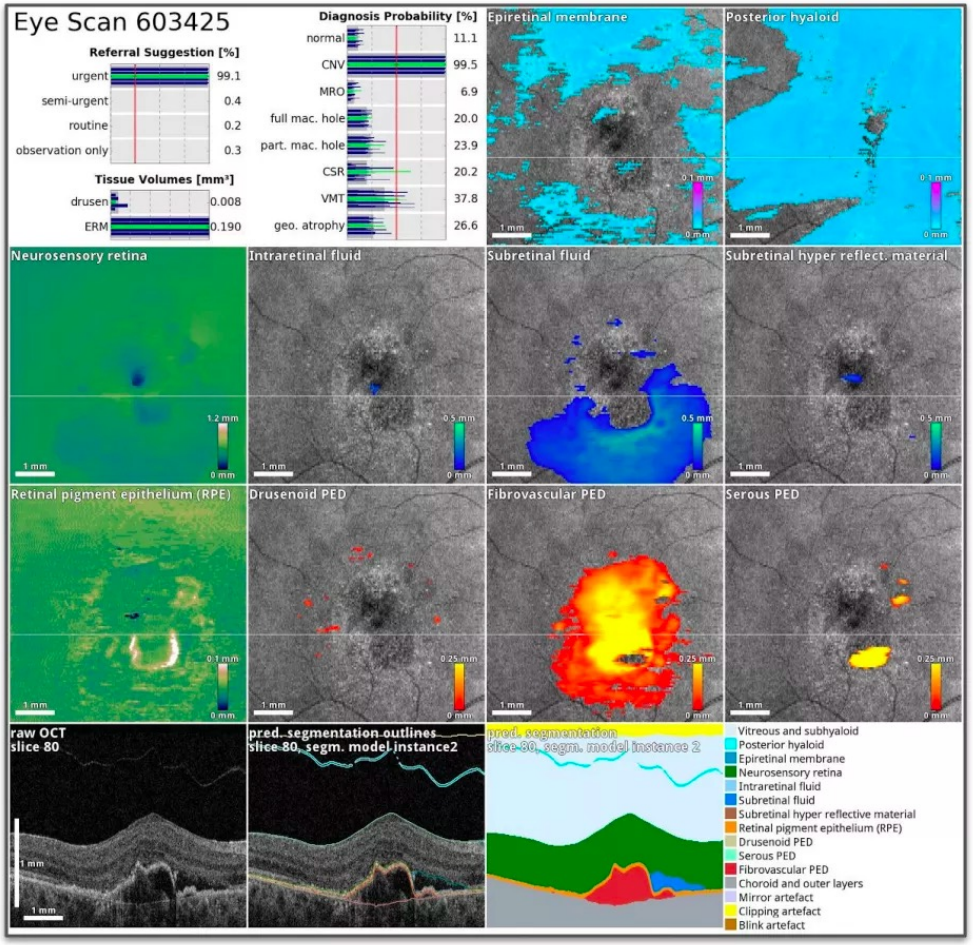
- **Medical imaging (MRI)**



http://images.nationalgeographic.com/wpf/media-live/photos/000/008/cache/brain-mri_848_990x742.jpg

The task, *T*

- **Medical imaging (Google Health)**



https://www.nature.com/articles/s41591-018-0107-6

The task, *T*

- **Fingerprint detection**

http://biometrics.mainguet.org/types/fingerprint/algo/whorl_loop_arche.jpg
http://biometrics.mainguet.org/types/fingerprint/algo/minutiae.jpg

The task, *T*

- Regression: The computer program is asked to predict a numerical value given some input.

- To solve this task, the learning algorithm is asked to output a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. This type of task is similar to classification, except that the format of output is different.

- An example of a regression task is the prediction of the expected claim amount that an insured person will make (used to set insurance premiums), or the prediction of future prices of securities. These kinds of predictions are also used for algorithmic trading.

The task, *T*

- **Fintech**

Disrupting (& Improving) Insurance

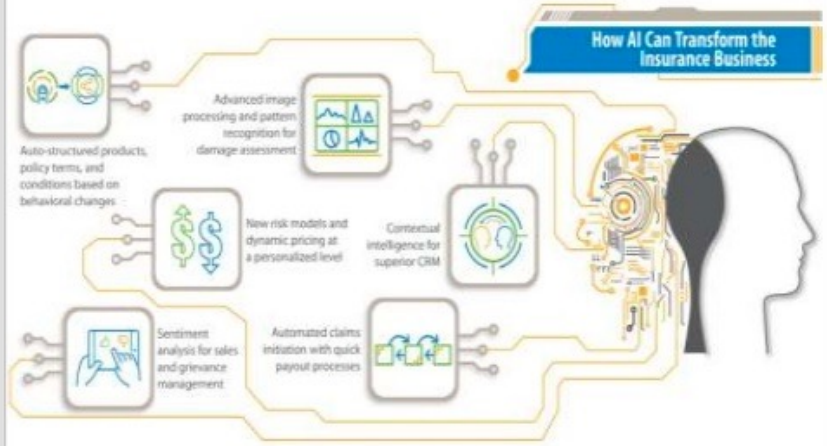
THE DIGITAL INSURER
The forum for digital insurance around the world
Join Our Group @Digitalinsurer

What does AI mean for Insurance?

- Robo Advisors, Virtual Assistants, AI enabled authentication
- Need Based Offering, Personalisation and Intelligent Pricing
- Predict Customer Needs and Behavior, Quick Decision Making, Optimize Sales Strategy
- Automate & Optimize Underwriting Model, Real-time Underwriting
- FNOL Damage Assessment, Fraud Detection, Assistance and Quick Processing

How AI can benefit Insurance

Operational efficiency → Revenue expansion → Customer experience → Competitive advantage



Source - http://www.tcs.com/SiteCollectionDocuments/White%20Papers/Artificial-intelligence-reimagines-insurance-0816-2.pdf

https://image.slidesharecdn.com/aiininsurance-primer-draft-version2-170327114347/95/ai-in-insurance-inview-primer-5-1024.jpg?cb=1490615256

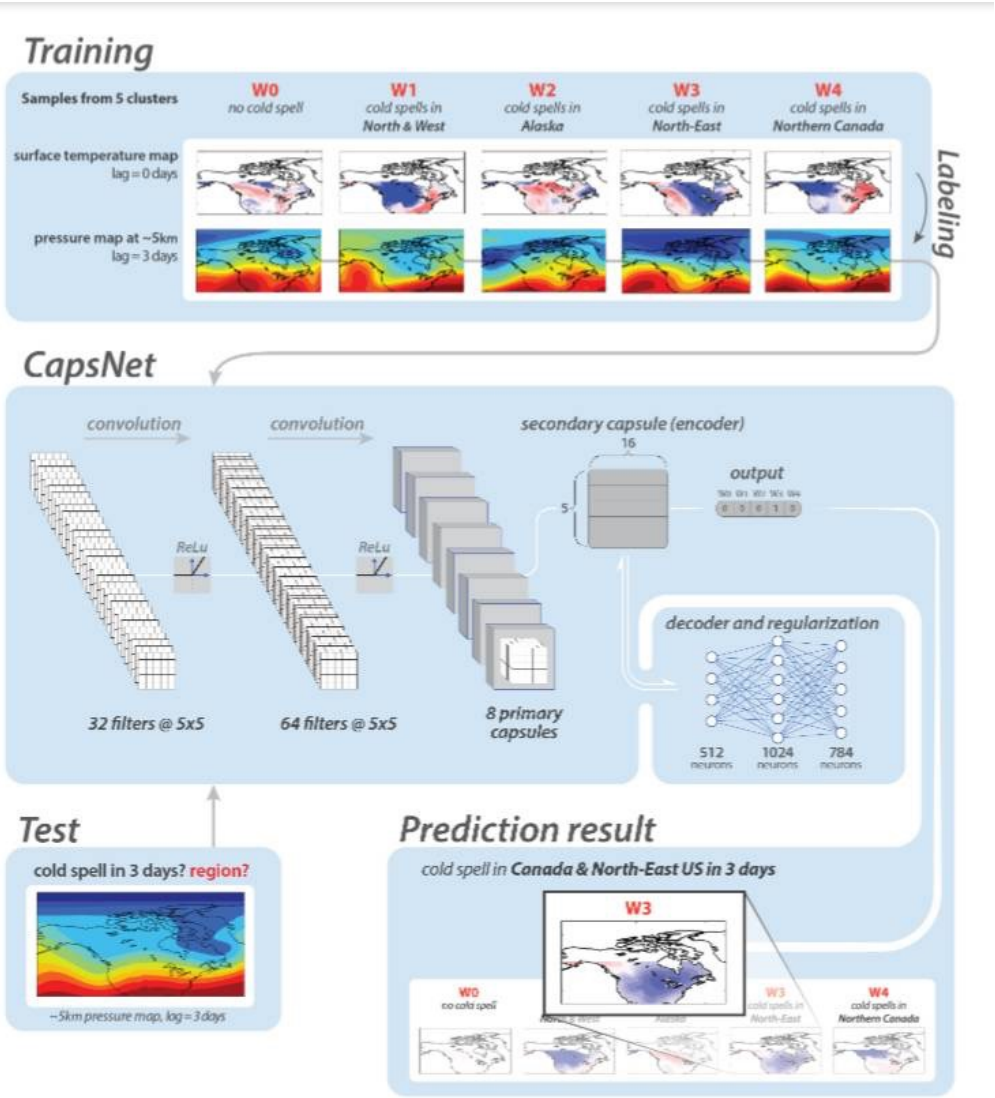
The task, *T*

- **Fintech (automatic trading)**

https://upload.wikimedia.org/wikipedia/commons/7/71/Forex.jpg

The task, *T*

- **Earth science (e.g., weather prediction)**



https://insidehpc.com/2020/02/deep-learning-for-predicting-severe-weather/

The task, *T*

- Transcription: The machine learning system is asked to observe a relatively unstructured representation of some kind of data and transcribe the information into discrete textual form.

- For example, in optical character recognition, the computer program is shown a photograph containing an image of text and is asked to return this text in the form of a sequence of characters (e.g., in ASCII or Unicode format). Google Street View uses deep learning to process address numbers in this way.

The task, *T*

- **Google street view**

https://gizmodo.com/google-built-an-insane-neural-network-to-id-house-numbe-1499897368

The task, *T*

- Another example is speech recognition, where the computer program is provided an audio waveform and emits a sequence of characters or word ID codes describing the words that were spoken in the audio recording.

- Deep learning is a crucial component of modern speech recognition systems used at major companies, including Microsoft, IBM and Google.

https://cdn1.i-scmp.com/sites/default/files/styles/landscape/public/2016/03/18/shutterstock_381624613.jpg?itok=rm2sFVDf

The task, *T*

- Machine translation: The input already consists of a sequence of symbols in some language, and the computer program must convert this into a sequence of symbols in another language.

- This is commonly applied to natural languages, such as translating from English to French.

The task, *T*

- Structured output: Structured output tasks involve any task where the output is a vector (or other data structure containing multiple values) with important relationships between the different elements.

- This is a broad category and subsumes the transcription and translation tasks described above, as well as many other tasks.

- One example is parsing—mapping a natural language sentence into a tree that describes its grammatical structure by tagging nodes of the trees as being verbs, nouns, adverbs, and so on.

The task, *T*

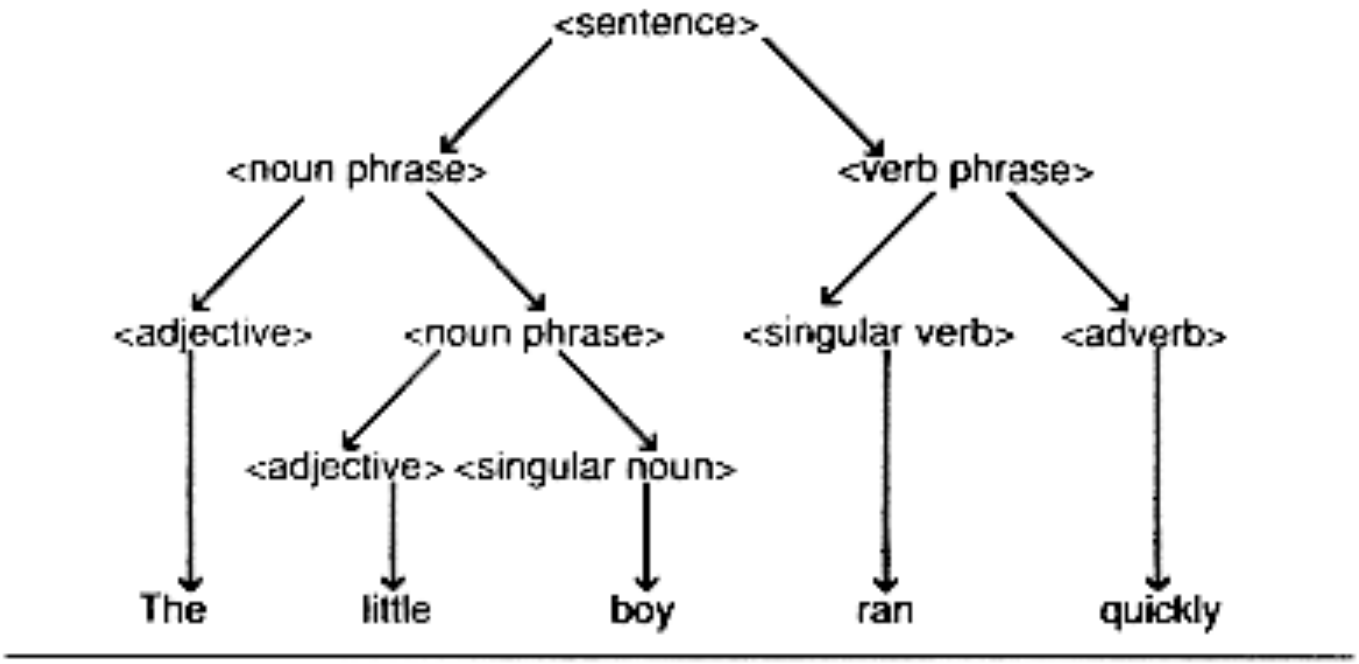
- **Parsing**

EXAMPLE 2 Finding the structure of a sentence

Using the grammar above, the sentence:

The little boy ran quickly

can be diagramed :



https://cstheory.stackexchange.com/questions/39859/visualizing-the-parse-structure-of-a-range-concatenation-grammar/39875

The task, *T*

- Another example is pixel-wise segmentation of images, where the computer program assigns every pixel in an image to a specific category.
- For example, deep learning can be used to annotate the locations of roads in aerial photographs.

https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef

The task, *T*

- The output form need not mirror the structure of the input as closely as in these annotation-style tasks. For example, in image captioning, the computer program observes an image and outputs a natural language sentence describing the image.

- These are called structured output tasks because the program must output several values that are all tightly interrelated. For example, the words produced by an image captioning program must form a valid sentence.

The task, *T*

- **Image/video captioning**

A young boy is playing basketball. 	Two dogs play in the grass. 	A dog swims in the water. 
A group of people walking down a street. 	A group of women dressed in formal attire. 	Two children play in the water. 
A skier is skiing down a snowy hill. 	A little girl in a pink shirt is swinging. 	A dog jumps over a hurdle. 

https://daniel.lasiman.com/post/image-captioning/

The task, *T*

- Anomaly detection: The computer program sifts through a set of events or objects and flags some of them as being unusual or atypical.

- An example of an anomaly detection task is credit card fraud detection. By modeling your purchasing habits, a credit card company can detect misuse of your cards. If a thief steals your credit card or credit card information, the thief's purchases will often come from a different probability distribution over purchase types than your own. The credit card company can prevent fraud by placing a hold on an account as soon as that card has been used for an uncharacteristic purchase.

The task, *T*

- Synthesis and sampling: The machine learning algorithm is asked to generate new examples that are similar to those in the training data.

- Synthesis and sampling via machine learning can be useful for media applications when generating large volumes of content by hand would be expensive, boring, or require too much time.

- For example, video games can automatically generate textures for large objects or landscapes, rather than requiring an artist to manually label each pixel.

The task, *T*

- **Art creation**

Yating
(雅婷)

https://www.electronicproducts.com/uploadedImages/Programming/Software/aipainter.jpg
https://s.mxmcdn.net/images-storage/albums4/4/5/1/7/6/9/37967154_350_350.jpg

The task, *T*

- **Image/video generation**

https://miro.medium.com/max/2468/1*Yw2KxjmIkj8yqS-ykLCQCQ.png

The task, *T*

Deepfake

https://www.abc.net.au/news/2018-09-27/fake-news-part-one/10308638?nw=0

The task, *T*

- In some cases, we want the sampling or synthesis procedure to generate a specific kind of output given the input. For example, in a speech synthesis task, we provide a written sentence and ask the program to emit an audio waveform containing a spoken version of that sentence.

- This is a kind of structured output task, but with the added qualification that there is no single correct output for each input, and we explicitly desire a large amount of variation in the output, in order for the output to seem more natural and realistic.

The task, *T*

- Imputation of missing values: The machine learning algorithm is given a new example $\boldsymbol{x} \in \mathbb{R}^n$, but with some entries x_i of $\boldsymbol{x}$ missing. The algorithm must provide a prediction of the values of the missing entries.

The task, *T*

- Denoising: The machine learning algorithm is given in input a corrupted example $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ obtained by an unknown corruption process from a clean example $\boldsymbol{x} \in \mathbb{R}^n$.

- The learner must predict the clean example $\boldsymbol{x}$ from its corrupted version $\tilde{\boldsymbol{x}}$, or more generally predict the conditional probability distribution $p(\boldsymbol{x}|\tilde{\boldsymbol{x}})$.

The task, *T*

- **Denoising**

http://people.tuebingen.mpg.de/burger/neural_denoising/

The task, *T*

- Density estimation or probability mass function estimation: The machine learning algorithm is asked to learn a function $p_{\text{model}}: \mathbb{R}^n \rightarrow \mathbb{R}$, where $p_{\text{model}}(\boldsymbol{x})$ can be interpreted as a probability density function (if $\boldsymbol{x}$ is continuous) or a probability mass function (if $\boldsymbol{x}$ is discrete) on the space that the examples were drawn from.

- To do such a task well, the algorithm needs to learn the structure of the data it has seen. It must know where examples cluster tightly and where they are unlikely to occur.

The task, *T*

- Most of the tasks described above require the learning algorithm to at least implicitly capture the structure of the probability distribution. Density estimation enables us to explicitly capture that distribution. In principle, we can then perform computations on that distribution to solve the other tasks as well.

- For example, if we have performed density estimation to obtain a probability distribution $p(\boldsymbol{x})$, we can use that distribution to solve the missing value imputation task. If a value x_i is missing, and all the other values, denoted $\boldsymbol{x}_{-i}$, are given, then we know the distribution over it is given by $p(x_i|\boldsymbol{x}_{-i})$. In practice, density estimation does not always enable us to solve all these related tasks, because in many cases the required operations on $p(\boldsymbol{x})$ are computationally intractable.

The performance measure, P

- To evaluate the abilities of a machine learning algorithm, we must design a quantitative measure of its performance. Usually this performance measure P is specific to the task T.

- For tasks such as classification, classification with missing inputs, and transcription, we often measure the accuracy of the model. Accuracy is just the proportion of examples for which the model produces the correct output.

- We can also obtain equivalent information by measuring the error rate, the proportion of examples for which the model produces an incorrect output.

The performance measure, P

- For tasks such as density estimation, it does not make sense to measure accuracy or error rate. Instead, we must use a different performance metric that gives the model a continuous-valued score for each example. The most common approach is to report the average log-probability the model assigns to some examples.

- Usually we are interested in how well the machine learning algorithm performs on data that it has not seen before, since this determines how well it will work when deployed in the real world. We therefore evaluate these performance measures using a test set of data that is separate from the data used for training the machine learning system.

The performance measure, P

- It is often difficult to choose a performance measure that corresponds well to the desired behavior of the system. In some cases, this is because it is difficult to decide what should be measured.

- For example, when performing a transcription task, should we measure the accuracy of the system at transcribing entire sequences, or should we use a more fine-grained performance measure that gives partial credit for getting some elements of the sequence correct?

- When performing a regression task, should we penalize the system more if it frequently makes medium-sized mistakes or if it rarely makes very large mistakes?

The performance measure, P

- In other cases, we know what quantity we would ideally like to measure, but measuring it is impractical.
- For example, this arises frequently in the context of density estimation. Many of the best probabilistic models represent probability distributions only implicitly. Computing the actual probability value assigned to a specific point in space in many such models is intractable.
- In these cases, one must design an alternative criterion that still corresponds to the design objectives, or design a good approximation to the desired criterion.

The experience, E

- Machine learning algorithms can be broadly categorized as unsupervised or supervised by what kind of experience they are allowed to have during the learning process.

- Most of the learning algorithms in this book can be understood as being allowed to experience an entire dataset. A dataset is a collection of many examples. Sometimes we call examples data points.

The experience, E

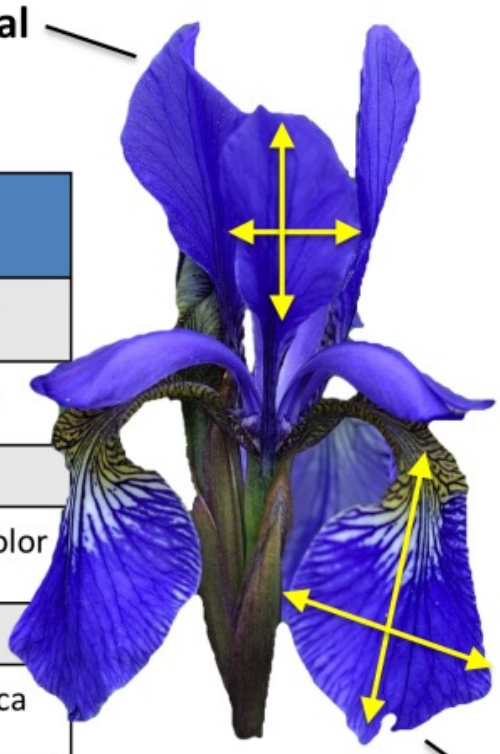
- One of the oldest datasets studied by statisticians and machine learning researchers is the Iris dataset.

- It is a collection of measurements of different parts of 150 iris plants. Each individual plant corresponds to one example. The features within each example are the measurements of each part of the plant: the sepal length, sepal width, petal length and petal width. The dataset also records which species each plant belonged to. Three different species are represented in the dataset.

The experience, E

Samples
(instances, observations)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica



Features
(attributes, measurements, dimensions)

Class labels
(targets)

The experience, E

- Unsupervised learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset.

- In the context of deep learning, we usually want to learn the entire probability distribution that generated a dataset, whether explicitly, as in density estimation, or implicitly, for tasks like synthesis or denoising.

- Some other unsupervised learning algorithms perform other roles, like clustering, which consists of dividing the dataset into clusters of similar examples.

The experience, E

- Supervised learning algorithms experience a dataset containing features, but each example is also associated with a label or target.

- For example, the Iris dataset is annotated with the species of each iris plant. A supervised learning algorithm can study the Iris dataset and learn to classify iris plants into three different species based on their measurements.

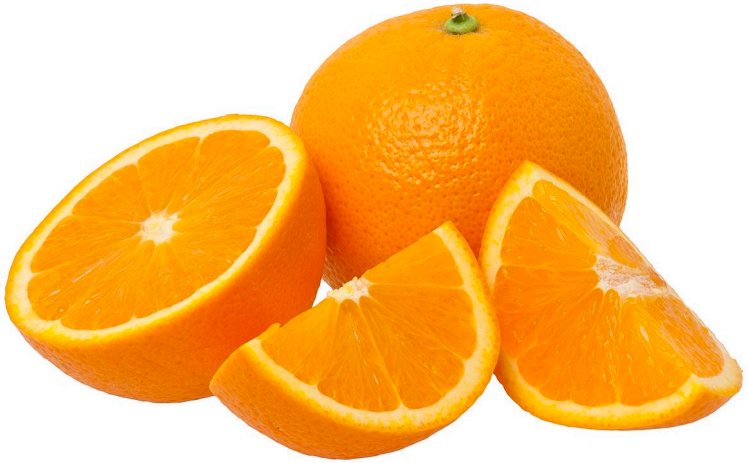
The experience, E

- Unsupervised learning involves observing several examples of a random vector $\mathbf{x}$ and attempting to implicitly or explicitly learn the probability distribution $p(\mathbf{x})$, or some interesting properties of that distribution.

- Supervised learning involves observing several examples of a random vector $\mathbf{x}$ and an associated value or vector $\mathbf{y}$, then learning to predict $\mathbf{y}$ from $\mathbf{x}$, usually by estimating $p(\mathbf{y}|\mathbf{x})$.

- The term supervised learning originates from the view of the target $\mathbf{y}$ being provided by an instructor or teacher who shows the machine learning system what to do. In unsupervised learning, there is no instructor or teacher, and the algorithm must learn to make sense of the data without this guide.

The experience, *E*



The experience, E

- Unsupervised learning and supervised learning are not formally defined terms. The lines between them are often blurred. Many machine learning technologies can be used to perform both tasks.

- For example, the chain rule of probability states that for a vector $\mathbf{x} \in \mathbb{R}^n$, the joint distribution can be decomposed as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(\mathrm{x}_i \mid \mathrm{x}_1, \ldots, \mathrm{x}_{i-1}). \tag{5.1}$$

This decomposition means that we can solve the ostensibly unsupervised problem of modeling $p(\mathbf{x})$ by splitting it into n supervised learning problems.

The experience, E

- Alternatively, we can solve the supervised learning problem of learning $p(y|\mathbf{x})$ by using traditional unsupervised learning technologies to learn the joint distribution $p(\mathbf{x}, y)$, then inferring

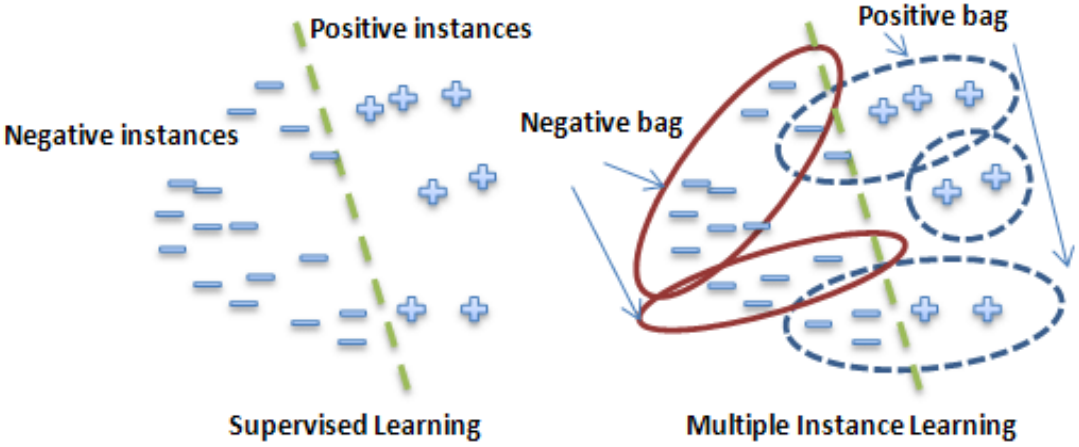
$$p(y \mid \mathbf{x}) = \frac{p(\mathbf{x}, y)}{\sum_{y'} p(\mathbf{x}, y')}. \tag{5.2}$$

- Traditionally, people refer to regression, classification and structured output problems as supervised learning.

- Density estimation in support of other tasks is usually considered unsupervised learning.

The experience, E

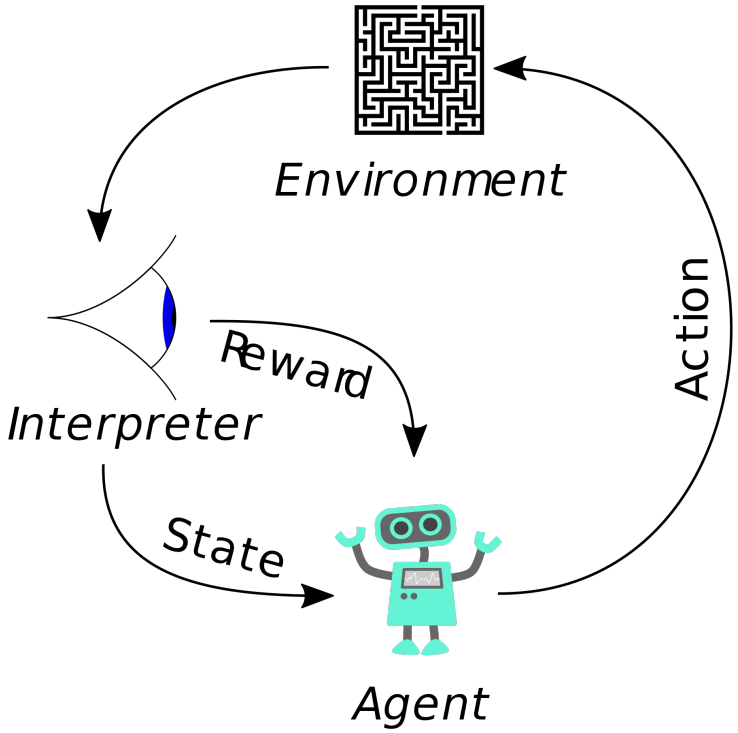
- Other variants of the learning paradigm are possible.
- For example, in semi-supervised learning, some examples include a supervision target but others do not.
- In multi-instance learning, an entire collection of examples is labeled as containing or not containing an example of a class, but the individual members of the collection are not labeled.



https://www.researchgate.net/figure/Comparison-of-supervised-learning-and-multi-instance-learning-The-classifier-is-learned_fig1_224265580

The experience, E

- Some machine learning algorithms do not just experience a fixed dataset. For example, reinforcement learning algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences.



https://zh.wikipedia.org/wiki/%E5%BC%BA%E5%8C%96%E5%AD%A6%E4%B9%A0#/media/File:Reinforcement_learning_diagram.svg

The experience, E

- Most machine learning algorithms simply experience a dataset. A dataset can be described in many ways. In all cases, a dataset is a collection of examples, which are in turn collections of features.

- One common way of describing a dataset is with a design matrix. A design matrix is a matrix containing a different example in each row. Each column of the matrix corresponds to a different feature. For instance, the Iris dataset contains 150 examples with four features for each example. This means we can represent the dataset with a design matrix $\boldsymbol{X} \in \mathbb{R}^{150\times4}$, where $X_{i,1}$ is the sepal length of plant i, $X_{i,2}$ is the sepal width of plant i, etc.

The experience, E

- To describe a dataset as a design matrix, it must be possible to describe each example as a vector, and each of these vectors must be the same size. This is not always possible.

- For example, if you have a collection of photographs with different widths and heights, then different photographs will contain different numbers of pixels, so not all the photographs may be described with the same length of vector.

- In cases like these, rather than describing the dataset as a matrix with m rows, we describe it as a set containing m elements: $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m)}\}$. This notation does not imply that any two example vectors $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$ have the same size.

The experience, E

- In the case of supervised learning, the example contains a label or target as well as a collection of features.

- For example, if we want to use a learning algorithm to perform object recognition from photographs, we need to specify which object appears in each of the photos. We might do this with a numeric code, with 0 signifying a person, 1 signifying a car, 2 signifying a cat, and so forth. When working with a dataset containing a design matrix of feature observations $\mathbf{X}$, we also provide a vector of labels $\boldsymbol{y}$, with y_i providing the label for example i.

- The label may be more than just a single number. For example, if we want to train a speech recognition system to transcribe entire sentences, then the label for each example sentence is a sequence of words.

Example: linear regression

- The goal of linear regression is to build a system that can take a vector $\boldsymbol{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}$ as its output. The output of linear regression is a linear function of the input.

- Let $\hat{y}$ be the value that our model predicts y should take on. We define the output to be

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}, \tag{5.3}$$

where $\boldsymbol{\omega} \in \mathbb{R}^n$ is a vector of parameters.

Example: linear regression

- ω_i is the coefficient that we multiply by feature x_i before summing up the contributions from all the features.

- We can think of $\boldsymbol{\omega}$ as a set of weights that determine how each feature affects the prediction. If a feature x_i receives a positive weight ω_i, then increasing the value of that feature increases the value of our prediction $\hat{y}$. If a feature receives a negative weight, then increasing the value of that feature decreases the value of our prediction.

- If a feature's weight is large in magnitude, then it has a large effect on the prediction. If a feature's weight is zero, it has no effect on the prediction.

Example: linear regression

- We thus have a definition of our task T : to predict y from $\boldsymbol{x}$ by outputting $\hat{y} = \boldsymbol{\omega}^T \boldsymbol{x}$.

- Next we need a definition of our performance measure, P.

- Suppose that we have a design matrix of m example inputs that we will not use for training, only for evaluating how well the model performs. We also have a vector of regression targets providing the correct value of y for each of these examples. Because this dataset will only be used for evaluation, we call it the test set. We refer to the design matrix of inputs as $\boldsymbol{X}^{(\text{test})}$ and the vector of regression targets as $\boldsymbol{y}^{(\text{test})}$.

Example: linear regression

- One way of measuring the performance of the model is to compute the mean squared error of the model on the test set. If $\hat{\boldsymbol{y}}^{(\text{test})}$ gives the predictions of the model on the test set, then the mean squared error is given by

$$\text{MSE}_{\text{test}} = \frac{1}{m}\sum_i (\hat{\boldsymbol{y}}^{(\text{test})} - \boldsymbol{y}^{(\text{test})})_i^2. \tag{5.4}$$

- One can see that this error measure decreases to 0 when $\hat{\boldsymbol{y}}^{(\text{test})} = \boldsymbol{y}^{(\text{test})}$. We can also see that

$$\text{MSE}_{\text{test}} = \frac{1}{m}||\hat{\boldsymbol{y}}^{(\text{test})} - \boldsymbol{y}^{(\text{test})}||_2^2, \tag{5.5}$$

  so the error increases whenever the Euclidean distance between the predictions and the targets increases.

Example: linear regression

- To make a machine learning algorithm, we need to design an algorithm that will improve the weights $\boldsymbol{\omega}$ in a way that reduces $\mathrm{MSE}_{\mathrm{test}}$ when the algorithm is allowed to gain experience by observing a training set $(\boldsymbol{X}^{(\mathrm{train})}, \boldsymbol{y}^{(\mathrm{train})})$.

- One intuitive way of doing this is just to minimize the mean squared error on the training set, $\mathrm{MSE}_{\mathrm{train}}$.

Example: linear regression

- To minimize $\mathrm{MSE}_{\mathrm{train}}$, we can simply solve for where its gradient is 0:

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}_{\mathrm{train}} = 0 \tag{5.6}$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \frac{1}{m} ||\hat{\boldsymbol{y}}^{(\mathrm{train})} - \boldsymbol{y}^{(\mathrm{train})}||_2^2 = 0 \tag{5.7}$$

$$\Rightarrow \frac{1}{m} \nabla_{\boldsymbol{w}} ||\boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})}||_2^2 = 0 \tag{5.8}$$

$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})} \right)^\top \left(\boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - \boldsymbol{y}^{(\mathrm{train})} \right) = 0 \tag{5.9}$$

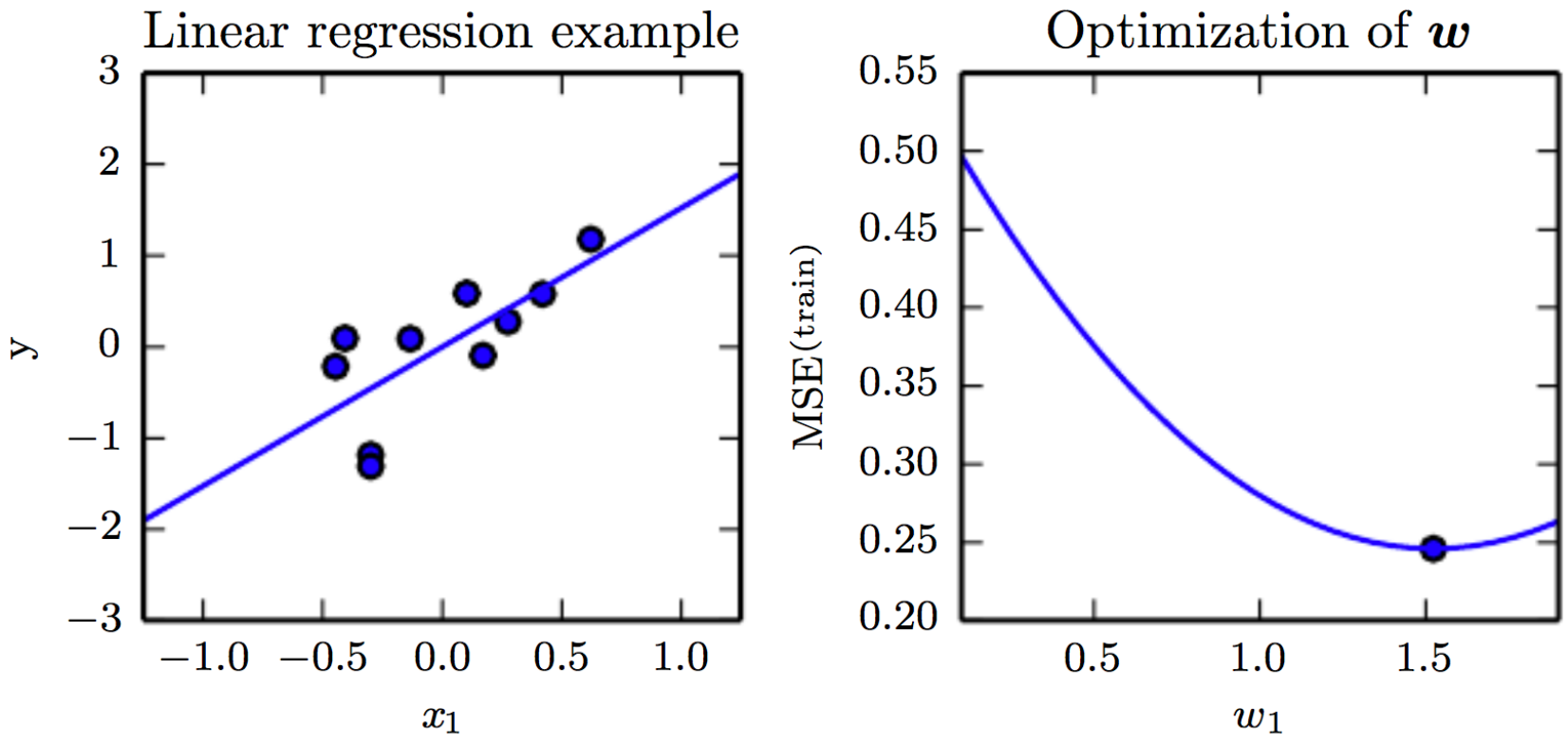
$$\Rightarrow \nabla_{\boldsymbol{w}} \left(\boldsymbol{w}^\top \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2 \boldsymbol{w}^\top \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} + \boldsymbol{y}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} \right) = 0 \tag{5.10}$$

$$\Rightarrow 2 \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \boldsymbol{w} - 2 \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} = 0 \tag{5.11}$$

$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})} \right)^{-1} \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})} \tag{5.12}$$

Example: linear regression

- The system of equations whose solution is given by equation 5.12 is known as the normal equations. Evaluating equation 5.12 constitutes a simple learning algorithm.



Example: linear regression

- It is worth noting that the term linear regression is often used to refer to a slightly more sophisticated model with one additional parameter—an intercept term b. In this model

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b, \tag{5.13}$$

so the mapping from parameters to predictions is still a linear function but the mapping from features to predictions is now an affine function.

- This extension to affine functions means that the plot of the model's predictions still looks like a line, but it need not pass through the origin.

Example: linear regression

- Instead of adding the bias parameter b, one can continue to use the model with only weights but augment $\boldsymbol{x}$ with an extra entry that is always set to 1. The weight corresponding to the extra 1 entry plays the role of the bias parameter. We frequently use the term "linear" when referring to affine functions in this course.

- The intercept term b is often called the bias parameter of the affine transformation. This terminology derives from the point of view that the output of the transformation is biased toward being b in the absence of any input. This term is different from the idea of a statistical bias, in which a statistical estimation algorithm's expected estimate of a quantity is not equal to the true quantity.

Capacity, overfitting and underfitting

- The central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs—not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization.

- We can compute some error measure on the training set, called the training error; and we reduce this training error. So far, what we have described is simply an optimization problem.

- What separates machine learning from optimization is that we want the generalization error (test error) to be low as well. The generalization error is defined as the expected value of the error on a new input.

Capacity, overfitting and underfitting

- We typically estimate the generalization error of a machine learning model by measuring its performance on a test set of examples that were collected separately from the training set.

- In our linear regression example, we trained the model by minimizing the training error,

$$\frac{1}{m^{(\text{train})}}||\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{y}^{(\text{train})}||_2^2, \tag{5.14}$$

but we actually care about the test error,

$$\frac{1}{m^{(\text{test})}}||\boldsymbol{X}^{(\text{test})}\boldsymbol{w} - \boldsymbol{y}^{(\text{test})}||_2^2$$

Capacity, overfitting and underfitting

- How can we affect performance on the test set when we can observe only the training set? The field of statistical learning theory provides some answers.

- If the training and the test set are collected arbitrarily, there is indeed little we can do. If we are allowed to make some assumptions about how the training and test set are collected, then we can make some progress.

Capacity, overfitting and underfitting

- The training and test data are generated by a probability distribution over datasets called the data-generating process.

- We typically make a set of assumptions known collectively as the i.i.d. assumptions. These assumptions are that the examples in each dataset are independent from each other, and that the training set and test set are identically distributed, drawn from the same probability distribution as each other.

- The same distribution is then used to generate every train example and every test example. We call that shared underlying distribution the data-generating distribution, denoted p_{data}.

Capacity, overfitting and underfitting

- One immediate connection we can observe between training error and test error is that the expected training error of a randomly selected model is equal to the expected test error of that model.

- Suppose we have a probability distribution $p(\boldsymbol{x}, y)$ and we sample from it repeatedly to generate the training set and the test set. For some fixed value $\boldsymbol{\omega}$, the expected training set error is exactly the same as the expected test set error, because both expectations are formed using the same dataset sampling process. The only difference between the two conditions is the name we assign to the dataset we sample.

Capacity, overfitting and underfitting

- However, when we use a machine learning algorithm, we do not fix the parameters ahead of time, then sample both datasets. We sample the training set, then use it to choose the parameters to reduce training set error, then sample the test set.

- Under this process, the expected test error is greater than or equal to the expected value of training error.

- The factors determining how well a machine learning algorithm will perform are its ability to
  - Make the training error small.
  - Make the gap between training and test error small.

Capacity, overfitting and underfitting

- One way to control the capacity of a learning algorithm is by choosing its hypothesis space, the set of functions that the learning algorithm is allowed to select as being the solution.

- For example, the linear regression algorithm has the set of all linear functions of its input as its hypothesis space. We can generalize linear regression to include polynomials, rather than just linear functions, in its hypothesis space. Doing so increases the model's capacity.

Capacity, overfitting and underfitting

- A polynomial of degree 1 gives us the linear regression model

$$\hat{y} = b + wx. \tag{5.15}$$

- By introducing x^2 as another feature provided to the linear regression model, we can learn a model that is quadratic as a function of x:

$$\hat{y} = b + w_1 x + w_2 x^2. \tag{5.16}$$

Though this model implements a quadratic function of its input, the output is still a linear function of the parameters, so we can still use the normal equations to train the model in closed form.

- We can continue to add more powers of x as additional features, for example, to obtain a polynomial of degree 9:

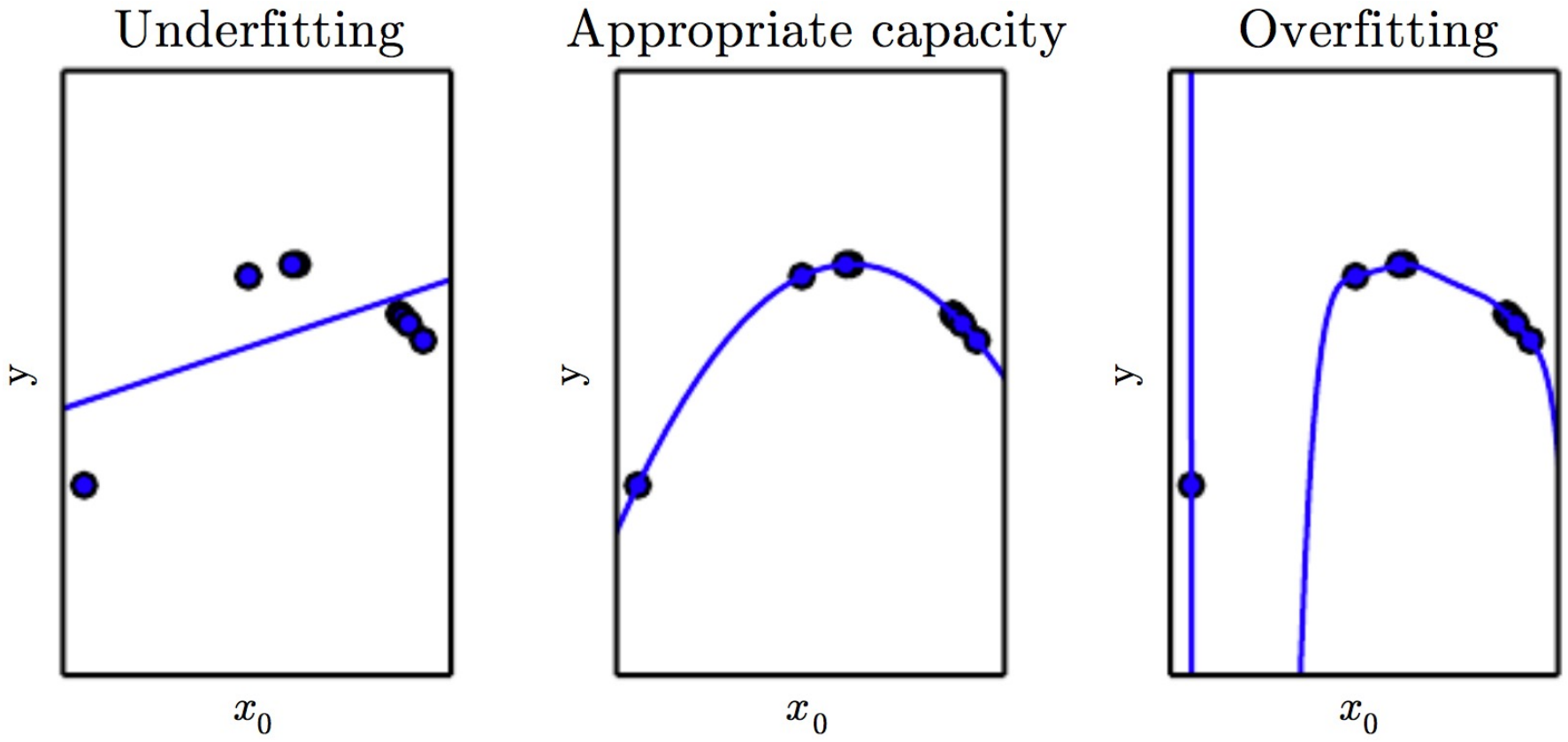
$$\hat{y} = b + \sum_{i=1}^{9} w_i x^i. \tag{5.17}$$

Capacity, overfitting and underfitting

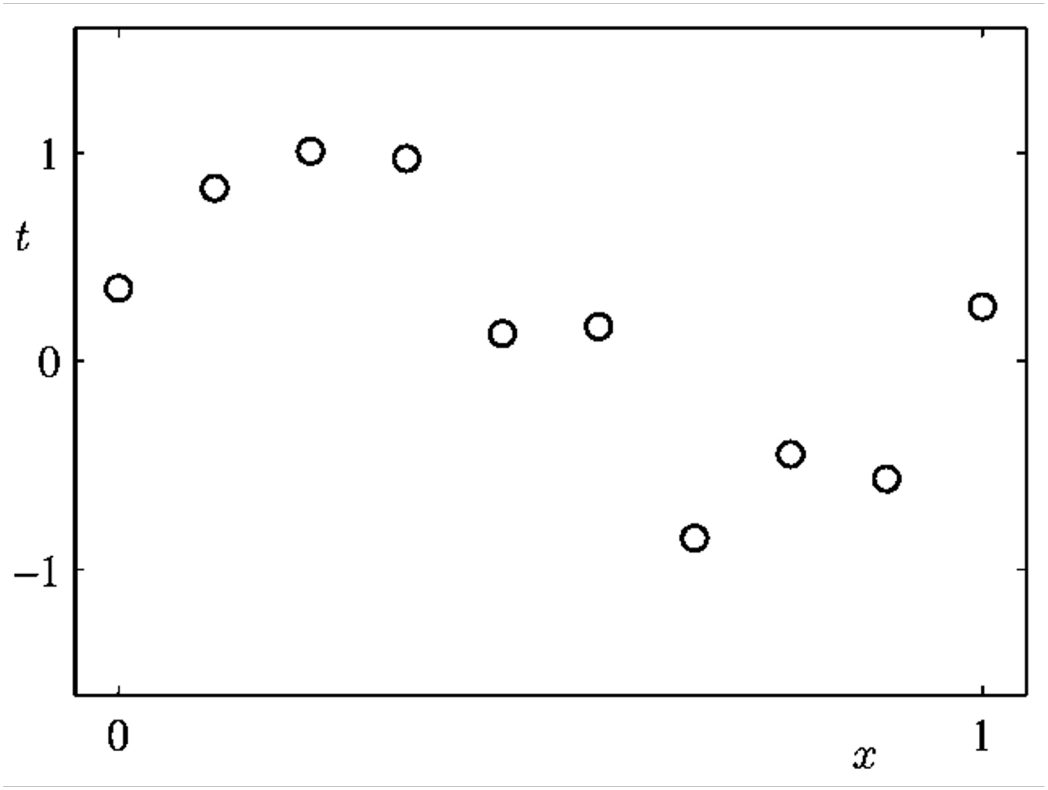
- Machine learning algorithms will generally perform best when their capacity is appropriate for the true complexity of the task they need to perform and the amount of training data they are provided with.

- Models with insufficient capacity are unable to solve complex tasks. Models with high capacity can solve complex tasks, but when their capacity is higher than needed to solve the present task, they may overfit.

Capacity, overfitting and underfitting

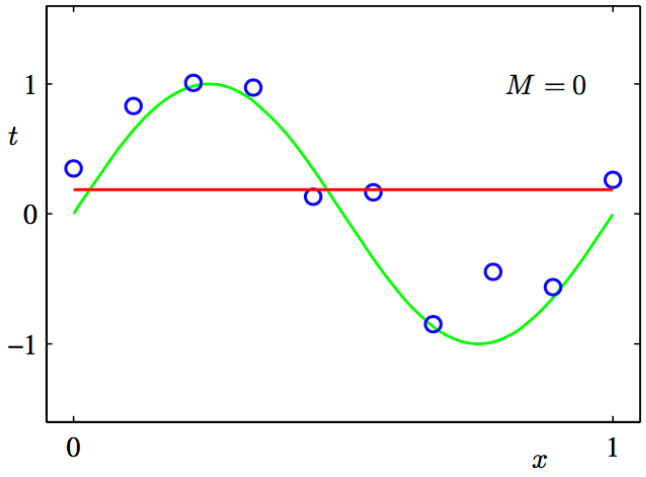


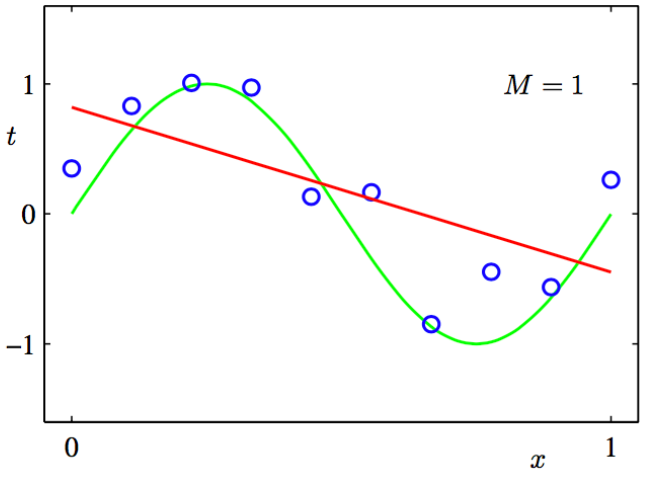
Capacity, overfitting and underfitting

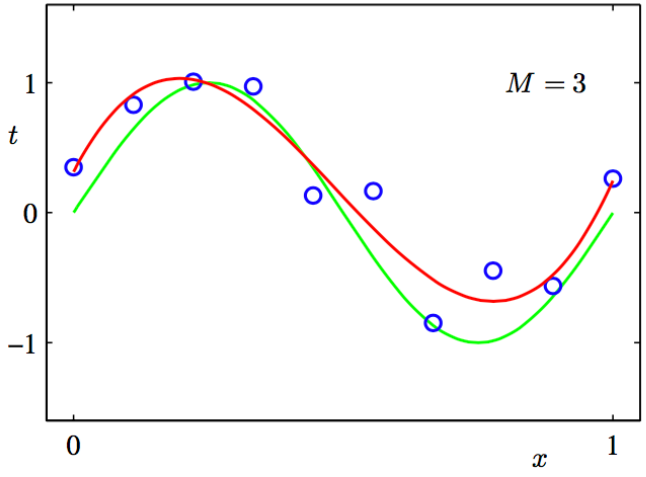


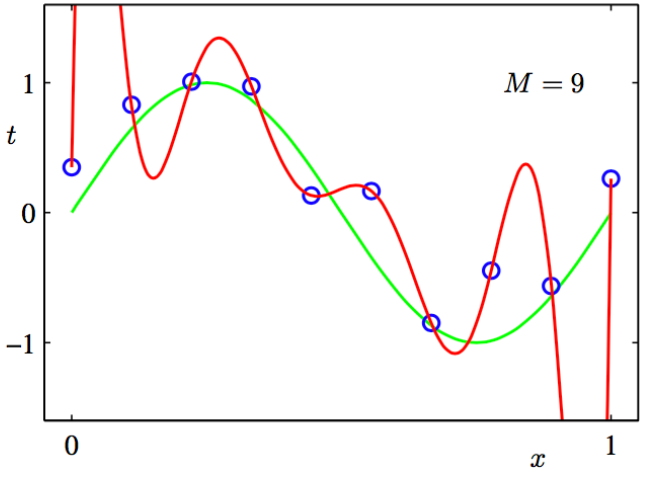
C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007

Capacity, overfitting and underfitting









Over-fitting

C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2007

Capacity, overfitting and underfitting

- The model specifies which family of functions the learning algorithm can choose from when varying the parameters in order to reduce a training objective. This is called the representational capacity of the model. In many cases, finding the best function within this family is a difficult optimization problem.

- Our modern ideas about improving the generalization of machine learning models are refinements of thought dating back to philosophers at least as early as Ptolemy. Occam's razor (c. 1287–1347) states that among competing hypotheses that explain known observations equally well, we should choose the "simplest" one.

Capacity, overfitting and underfitting

- Statistical learning theory provides various means of quantifying model capacity. The Vapnik-Chervonenkis (VC) dimension measures the capacity of a binary classifier. The VC dimension is defined as being the largest possible value of m for which there exists a training set of m different $\boldsymbol{x}$ points that the classifier can label arbitrarily.

- The most important results in statistical learning theory show that the discrepancy between training error and generalization error is bounded from above by a quantity that grows as the model capacity grows but shrinks as the number of training examples increases.

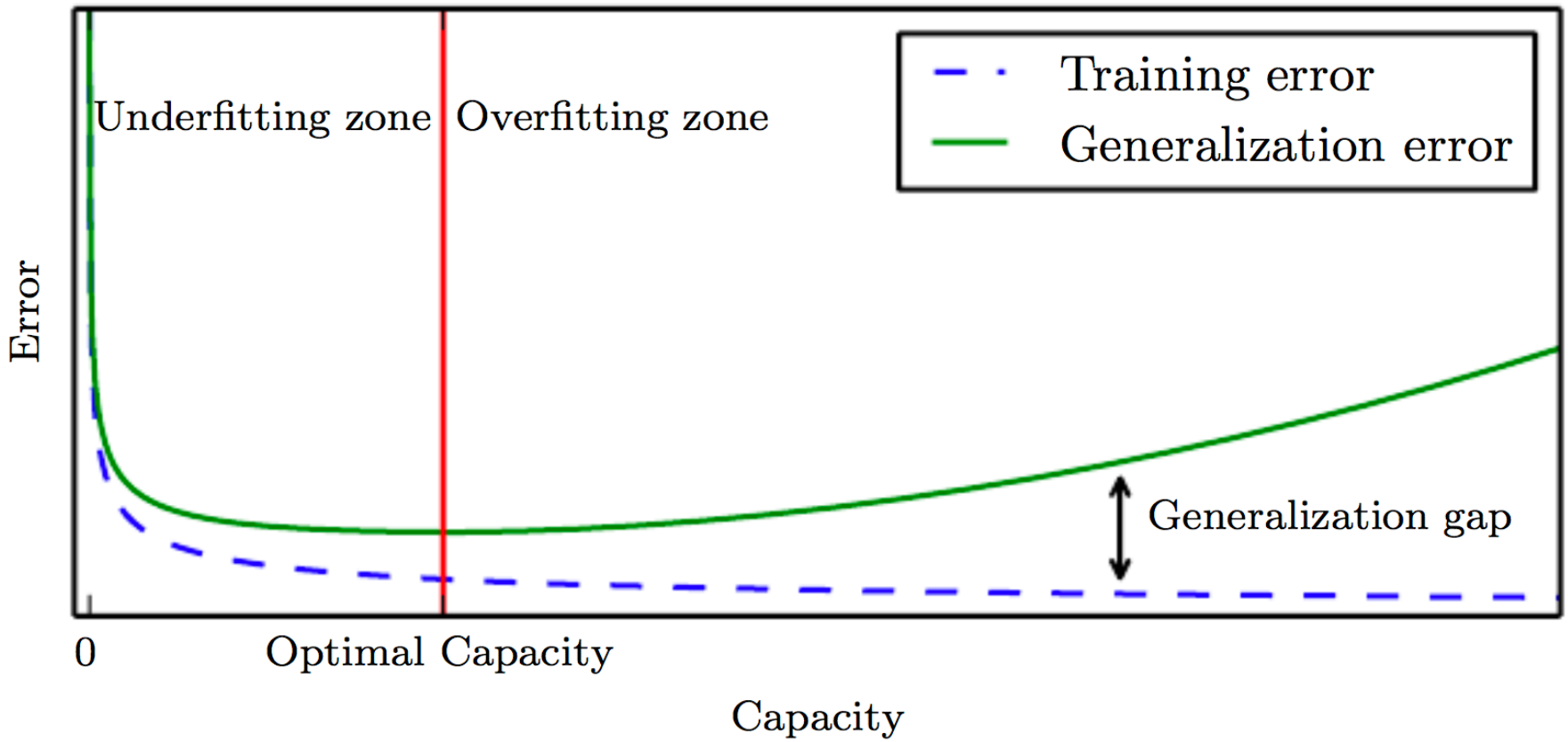
Capacity, overfitting and underfitting

- These bounds are rarely used in practice when working with deep learning algorithms. This is in part because the bounds are often quite loose and in part because it can be quite difficult to determine the capacity of deep learning algorithms.

- The problem of determining the capacity of a deep learning model is especially difficult because the effective capacity is limited by the capabilities of the optimization algorithm, and we have little theoretical understanding of the general nonconvex optimization problems involved in deep learning.

- While simpler functions are more likely to generalize, we must still choose a sufficiently complex hypothesis to achieve low training error.

Capacity, overfitting and underfitting



Capacity, overfitting and underfitting

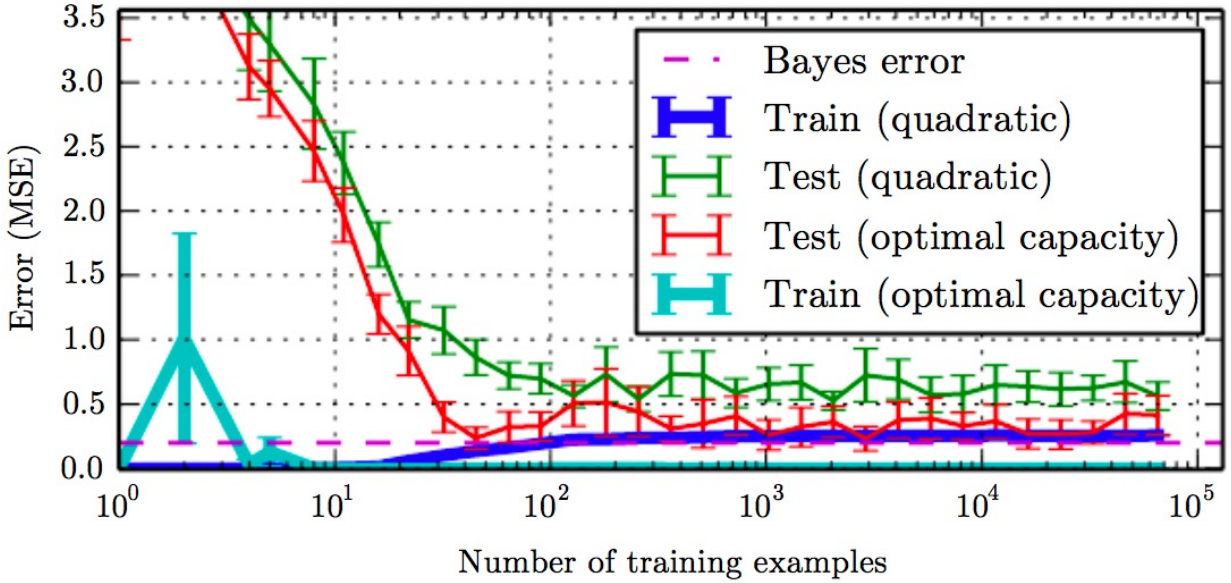
- The ideal model is an oracle that simply knows the true probability distribution that generates the data.
- Even such a model will still incur some error on many problems, because there may still be some noise in the distribution.
- In the case of supervised learning, the mapping from $\boldsymbol{x}$ to y may be inherently stochastic, or y may be a deterministic function that involves other variables besides those included in $\boldsymbol{x}$. The error incurred by an oracle making predictions from the true distribution $p(\boldsymbol{x}, y)$ is called the Bayes error.

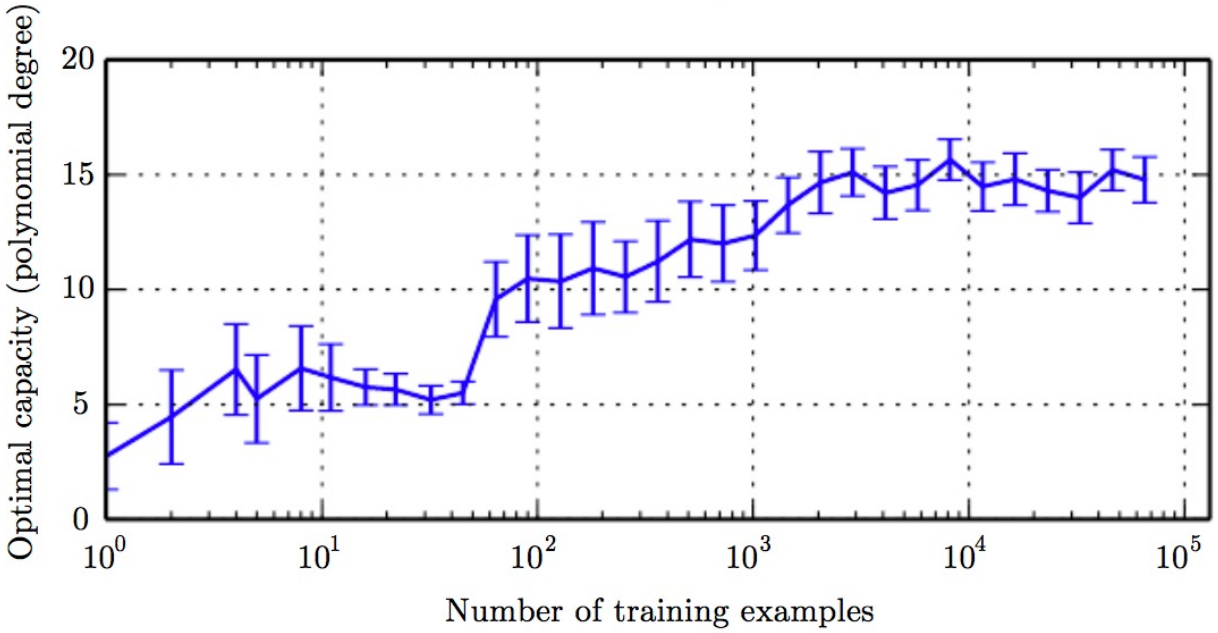
Capacity, overfitting and underfitting

- Training and generalization error vary as the size of the training set varies. Expected generalization error can never increase as the number of training examples increases.

- Note that it is possible for the model to have optimal capacity and yet still have a large gap between training and generalization errors. In this situation, we may be able to reduce this gap by gathering more training examples.

Capacity, overfitting and underfitting





The no free lunch theorem

- The no free lunch theorem for machine learning states that, averaged over all possible data-generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

- In other words, in some sense, no machine learning algorithm is universally any better than any other. The most sophisticated algorithm we can conceive of has the same average performance (over all possible tasks) as merely predicting that every point belongs to the same class.

The no free lunch theorem

- Fortunately, these results hold only when we average over all possible data-generating distributions. If we make assumptions about the kinds of probability distributions we encounter in real-world applications, then we can design learning algorithms that perform well on these distributions.

- This means that the goal of machine learning research is not to seek a universal learning algorithm or the absolute best learning algorithm. Instead, our goal is to understand what kinds of distributions are relevant to the "real world" that an AI agent experiences, and what kinds of machine learning algorithms perform well on data drawn from the kinds of data-generating distributions we care about.

Regularization

- The no free lunch theorem implies that we must design our machine learning algorithms to perform well on a specific task. We do so by building a set of preferences into the learning algorithm. When these preferences are aligned with the learning problems that we ask the algorithm to solve, it performs better.

- We can give a learning algorithm a preference for one solution over another in its hypothesis space. This means that both functions are eligible, but one is preferred. The unpreferred solution will be chosen only if it fits the training data significantly better than the preferred solution.

Regularization

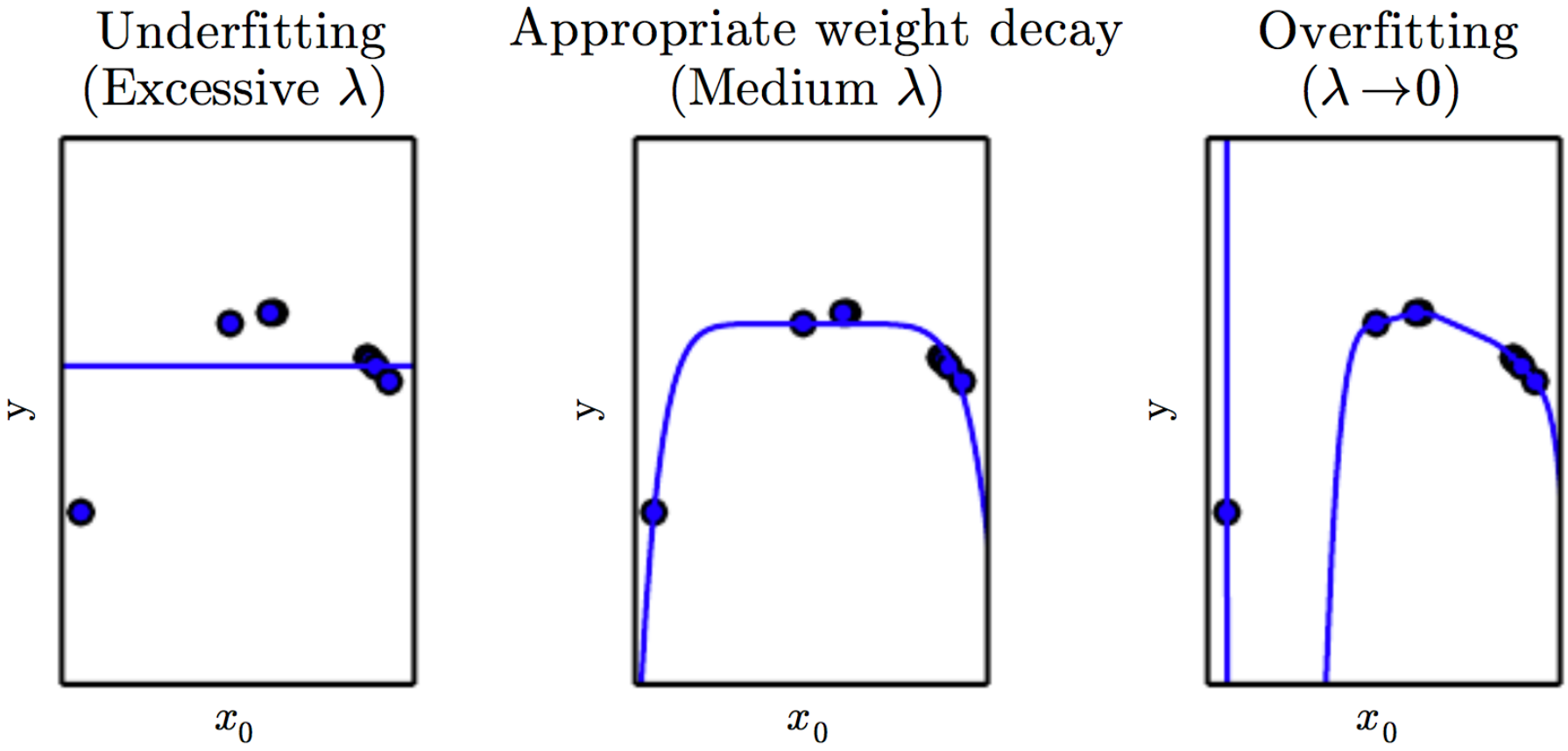
- For example, we can modify the training criterion for linear regression to include weight decay. We minimize a sum comprising both the mean squared error on the training and a criterion $J(\boldsymbol{\omega})$ that expresses a preference for the weights to have smaller squared L^2 norm:

$$J(\boldsymbol{w}) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{w}^\top \boldsymbol{w}, \tag{5.18}$$

  where λ is a value chosen ahead of time that controls the strength of our preference for smaller weights. When $\lambda = 0$, we impose no preference, and larger λ forces the weights to become smaller.

- Minimizing $J(\omega)$ results in a choice of weights that make a tradeoff between fitting the training data and being small. This gives us solutions that have a smaller slope, or that put weight on fewer of the features.

Regularization



Regularization

- More generally, we can regularize a model that learns a function $f(\boldsymbol{x}; \boldsymbol{\theta})$ by adding a penalty called a regularizer to the cost function. In the case of weight decay, the regularizer is $\Omega(\boldsymbol{\omega}) = \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}$.

- Expressing preferences for one function over another is a more general way of controlling a model's capacity than including or excluding members from the hypothesis space. We can think of excluding a function from a hypothesis space as expressing an infinitely strong preference against that function.

- Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

Regularization

- The no free lunch theorem has made it clear that there is no best machine learning algorithm, and, in particular, no best form of regularization. Instead we must choose a form of regularization that is well suited to the particular task we want to solve.

- The philosophy of deep learning in general and this book in particular is that a wide range of tasks may all be solved effectively using very general-purpose forms of regularization.

Hyperparameters and validation sets

- Most machine learning algorithms have hyperparameters, settings that we can use to control the algorithm's behavior. The values of hyperparameters are not adapted by the learning algorithm itself.

- The polynomial regression example has a single hyperparameter: the degree of the polynomial, which acts as a capacity hyperparameter. The λ value used to control the strength of weight decay is another example of a hyperparameter.

Hyperparameters and validation sets

- Sometimes a setting is chosen to be a hyperparameter that the learning algorithm does not learn because the setting is difficult to optimize.

- More frequently, the setting must be a hyperparameter because it is not appropriate to learn that hyperparameter on the training set. If learned on the training set, such hyperparameters would always choose the maximum possible model capacity, resulting in overfitting.

- To solve this problem, we need a validation set of examples that the training algorithm does not observe.

Hyperparameters and validation sets

- It is important that the test examples are not used in any way to make choices about the model, including its hyperparameters. For this reason, no example from the test set can be used in the validation set.

- Therefore, we always construct the validation set from the training data. Specifically, we split the training data into two disjoint subsets. One of these subsets is used to learn the parameters. The other subset is our validation set, used to estimate the generalization error during or after training, allowing for the hyperparameters to be updated accordingly.

- Typically, one uses about 80 percent of the training data for training and 20 percent for validation.

Cross-validation

- Dividing the dataset into a fixed training set and a fixed test set can be problematic if it results in the test set being small.

- An idea is repeating the training and testing computation on different randomly chosen subsets or splits of the original dataset. The most common of these is the k-fold cross-validation procedure, in which a partition of the dataset is formed by splitting it into k nonoverlapping subsets. The test error may then be estimated by taking the average test error across k trials. On trial i, the i-th subset of the data is used as the test set, and the rest of the data is used as the training set.