Chapter 5

Machine Learning Basics

Deep learning is a specific kind of machine learning. To un
well, one must have a solid understanding of the basic princip
This chapter provides a brief course in the most important
are applied throughout the rest of the book. Novice reader
wider perspective are encouraged to consider machine lear
more comprehensive coverage of the fundamentals, such as M
(2006). If you are already familiar with machine learning l
ahead to section 5.11. That section covers some perspectives
learning techniques that have strongly influenced the develo
algorithms.

We begin with a definition of what a learning algorit
example: the linear regression algorithm. We then procee

descent. We describe how to combine various algorithm an optimization algorithm, a cost function, a model, and machine learning algorithm. Finally, in section 5.11, we factors that have limited the ability of traditional machine These challenges have motivated the development of deep le overcome these obstacles.

5.1 Learning Algorithms

A machine learning algorithm is an algorithm that is ab But what do we mean by learning? Mitchell (1997) provide "A computer program is said to learn from experience E class of tasks T and performance measure P, if its perforn measured by P, improves with experience E." One can im experiences E, tasks T, and performance measures P, and this book to formally define what may be used for each of in the following sections, we provide intuitive description different kinds of tasks, performance measures, and experi to construct machine learning algorithms.

5.1.1 The Task, T

Machine learning enables us to tackle tasks that are too fixed programs written and designed by human beings. philosophical point of view, machine learning is interesting understanding of it entails developing our understanding underlie intelligence

Many kinds of tasks can be solved with machine learn common machine learning tasks include the following:

- **Classification**: In this type of task, the computer prog which of k categories some input belongs to. To solve algorithm is usually asked to produce a function $f : \mathbb{R}$ $y = f(\boldsymbol{x})$, the model assigns an input described by identified by numeric code y. There are other varian task, for example, where f outputs a probability di An example of a classification task is object recogn is an image (usually described as a set of pixel brigh output is a numeric code identifying the object in th the Willow Garage PR2 robot is able to act as a wai different kinds of drinks and deliver them to people fellow *et al.*, 2010). Modern object recognition is be deep learning (Krizhevsky *et al.*, 2012; Ioffe and Sz recognition is the same basic technology that enables faces (Taigman *et al.*, 2014), which can be used to au in photo collections and for computers to interact mo users.

- **Classification with missing inputs**: Classificatio lenging if the computer program is not guaranteed that its input vector will always be provided. To solve the learning algorithm only has to define a *single* function input to a categorical output. When some of the in rather than providing a single classification function, must learn a *set* of functions. Each function correspon

- **Regression**: In this type of task, the computer progra
numerical value given some input. To solve this task,
is asked to output a function $f : \mathbb{R}^n \to \mathbb{R}$. This typ
classification, except that the format of output is dif
a regression task is the prediction of the expected
insured person will make (used to set insurance premi
of future prices of securities. These kinds of predict
algorithmic trading.

- **Transcription**: In this type of task, the machine le
to observe a relatively unstructured representation
and transcribe the information into discrete textual
optical character recognition, the computer program
containing an image of text and is asked to return t
a sequence of characters (e.g., in ASCII or Unicode f
View uses deep learning to process address numbers i
et al., 2014d). Another example is speech recognitio
program is provided an audio waveform and emits a se
word ID codes describing the words that were spoken
Deep learning is a crucial component of modern spee
used at major companies, including Microsoft, IBM
et al., 2012b).

- **Machine translation**: In a machine translation ta
consists of a sequence of symbols in some language, and
must convert this into a sequence of symbols in ano
commonly applied to natural languages, such as trans
French. Deep learning has recently begun to have ar

For example, deep learning can be used to annotate
in aerial photographs (Mnih and Hinton, 2010). T
not mirror the structure of the input as closely as in
tasks. For example, in image captioning, the compute
image and outputs a natural language sentence descr
et al., 2014a,b; Mao *et al.*, 2015; Vinyals *et al.*, 2015b
Karpathy and Li, 2015; Fang *et al.*, 2015; Xu *et al*
are called *structured output tasks* because the progra
values that are all tightly interrelated. For example,
an image captioning program must form a valid sent

- **Anomaly detection**: In this type of task, the co
through a set of events or objects and flags some of
or atypical. An example of an anomaly detection ta
detection. By modeling your purchasing habits, a cre
detect misuse of your cards. If a thief steals your cre
information, the thief's purchases will often come from
distribution over purchase types than your own. Th
can prevent fraud by placing a hold on an account as
been used for an uncharacteristic purchase. See Chan
survey of anomaly detection methods.

- **Synthesis and sampling**: In this type of task, th
gorithm is asked to generate new examples that are
training data. Synthesis and sampling via machine
for media applications when generating large volum
would be expensive, boring, or require too much tim
games can automatically generate textures for large

missing. The algorithm must provide a prediction of t
entries.

- **Denoising**: In this type of task, the machine learnin
input a *corrupted example* $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ obtained by an unkn
from a *clean example* $\boldsymbol{x} \in \mathbb{R}^n$. The learner must pre
$\boldsymbol{x}$ from its corrupted version $\tilde{\boldsymbol{x}}$, or more generally p
probability distribution $p(\boldsymbol{x} \mid \tilde{\boldsymbol{x}})$.

- **Density estimation** or **probability mass functio**
density estimation problem, the machine learning algor
function $p_{\text{model}} : \mathbb{R}^n \to \mathbb{R}$, where $p_{\text{model}}(\boldsymbol{x})$ can be inte
density function (if $\mathbf{x}$ is continuous) or a probability
discrete) on the space that the examples were drawn f
well (we will specify exactly what that means when w
measures P), the algorithm needs to learn the structur
It must know where examples cluster tightly and whe
occur. Most of the tasks described above require the l
least implicitly capture the structure of the probabilit
estimation enables us to explicitly capture that dist
we can then perform computations on that distribu
tasks as well. For example, if we have performed densi
a probability distribution $p(\boldsymbol{x})$, we can use that di
missing value imputation task. If a value x_i is miss
values, denoted $\boldsymbol{x}_{-i}$, are given, then we know the dist
by $p(x_i \mid \boldsymbol{x}_{-i})$. In practice, density estimation does n
solve all these related tasks, because in many cases t
on $p(\boldsymbol{x})$ are computationally intractable.

proportion of examples for which the model produces the c also obtain equivalent information by measuring the **erro** of examples for which the model produces an incorrect out the error rate as the expected 0-1 loss. The 0-1 loss on a p if it is correctly classified and 1 if it is not. For tasks such it does not make sense to measure accuracy, error rate, or loss. Instead, we must use a different performance metric a continuous-valued score for each example. The most co report the average log-probability the model assigns to som

Usually we are interested in how well the machine learn on data that it has not seen before, since this determines how deployed in the real world. We therefore evaluate these perfo a **test set** of data that is separate from the data used for learning system.

The choice of performance measure may seem straight but it is often difficult to choose a performance measure th the desired behavior of the system.

In some cases, this is because it is difficult to decide wh For example, when performing a transcription task, should w of the system at transcribing entire sequences, or should we performance measure that gives partial credit for getting sequence correct? When performing a regression task, s system more if it frequently makes medium-sized mistake very large mistakes? These kinds of design choices depend

In other cases, we know what quantity we would ideall measuring it is impractical. For example, this arises frequ density estimation. Many of the best probabilistic models

to experience an entire **dataset**. A dataset is a collection
defined in section 5.1.1. Sometimes we call examples **data**

One of the oldest datasets studied by statisticians anc
searchers is the Iris dataset (Fisher, 1936). It is a collec
of different parts of 150 iris plants. Each individual pla
example. The features within each example are the meas
of the plant: the sepal length, sepal width, petal length
dataset also records which species each plant belonged to.
are represented in the dataset.

Unsupervised learning algorithms experience a da
features, then learn useful properties of the structure of this
of deep learning, we usually want to learn the entire proba
generated a dataset, whether explicitly, as in density estim
tasks like synthesis or denoising. Some other unsupervise
perform other roles, like clustering, which consists of div
clusters of similar examples.

Supervised learning algorithms experience a datas
but each example is also associated with a **label** or **target**
dataset is annotated with the species of each iris plant.
algorithm can study the Iris dataset and learn to classify
different species based on their measurements.

Roughly speaking, unsupervised learning involves obse
of a random vector $\mathbf{x}$ and attempting to implicitly or exp
bility distribution $p(\mathbf{x})$, or some interesting properties of t
supervised learning involves observing several examples of a
an associated value or vector $\mathbf{y}$, then learning to predict
estimating $p(\mathbf{y} \mid \mathbf{x})$. The term **supervised learning** origi

modeling $p(\mathbf{x})$ by splitting it into n supervised learning prob
can solve the supervised learning problem of learning $p(y \mid$
unsupervised learning technologies to learn the joint dist
inferring

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

Though unsupervised learning and supervised learning are
or distinct concepts, they do help roughly categorize some o
machine learning algorithms. Traditionally, people refer to r
and structured output problems as supervised learning.
support of other tasks is usually considered unsupervised l

Other variants of the learning paradigm are possible.
supervised learning, some examples include a supervision
not. In multi-instance learning, an entire collection of e
containing or not containing an example of a class, but th
of the collection are not labeled. For a recent example of r
with deep models, see Kotzias *et al.* (2015).

Some machine learning algorithms do not just experien
example, **reinforcement learning** algorithms interact wi
there is a feedback loop between the learning system and
algorithms are beyond the scope of this book. Please see Su
or Bertsekas and Tsitsiklis (1996) for information about r
and Mnih *et al.* (2013) for the deep learning approach to r

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

length of vector. In Section 9.7 and chapter 10, we describe types of such heterogeneous data. In cases like these, rath dataset as a matrix with m rows, we describe it as a set c $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m)}\}$. This notation does not imply that ar $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$ have the same size.

In the case of supervised learning, the example contai well as a collection of features. For example, if we want to u to perform object recognition from photographs, we need t appears in each of the photos. We might do this with a signifying a person, 1 signifying a car, 2 signifying a cat, an working with a dataset containing a design matrix of featu also provide a vector of labels $\boldsymbol{y}$, with y_i providing the labe

Of course, sometimes the label may be more than just example, if we want to train a speech recognition syster sentences, then the label for each example sentence is a sec

Just as there is no formal definition of supervised and there is no rigid taxonomy of datasets or experiences. The st cover most cases, but it is always possible to design new on

5.1.4 Example: Linear Regression

Our definition of a machine learning algorithm as an algo of improving a computer program's performance at some somewhat abstract. To make this more concrete, we pre simple machine learning algorithm: **linear regression**. example repeatedly as we introduce more machine learning understand the algorithm's behavior each feature affects the prediction. If a feature x_i receive then increasing the value of that feature increases the val If a feature receives a negative weight, then increasing th decreases the value of our prediction. If a feature's weight then it has a large effect on the prediction. If a feature's w effect on the prediction.

We thus have a definition of our task T: to predict y $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$. Next we need a definition of our performance m

Suppose that we have a design matrix of m example i use for training, only for evaluating how well the model p a vector of regression targets providing the correct value examples. Because this dataset will only be used for evalua set. We refer to the design matrix of inputs as $\boldsymbol{X}^{(\text{test})}$ and targets as $\boldsymbol{y}^{(\text{test})}$.

One way of measuring the performance of the model is **squared error** of the model on the test set. If $\hat{\boldsymbol{y}}^{(\text{test})}$ give model on the test set, then the mean squared error is give

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

Intuitively, one can see that this error measure decreases to We can also see that

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

so the error increases whenever the Euclidean distance be

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

$$\Rightarrow \nabla_{\boldsymbol{w}}\left(\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{y}^{(\text{train})}\right)^\top\left(\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{y}\right.$$

$$\Rightarrow \nabla_{\boldsymbol{w}}\left(\boldsymbol{w}^\top\boldsymbol{X}^{(\text{train})\top}\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - 2\boldsymbol{w}^\top\boldsymbol{X}^{(\text{train})\top}\boldsymbol{y}^{(\text{train})} +\right.$$

$$\Rightarrow 2\boldsymbol{X}^{(\text{train})\top}\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - 2\boldsymbol{X}^{(\text{train})\top}\boldsymbol{y}^{(\text{trai}}$$

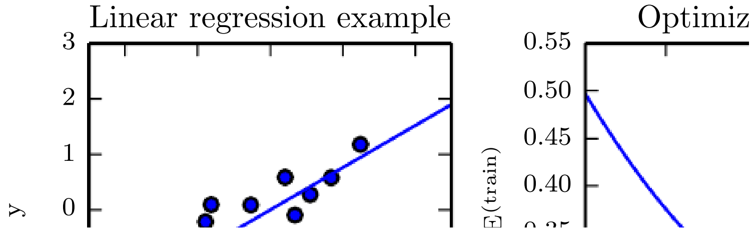
$$\Rightarrow \boldsymbol{w} = \left(\boldsymbol{X}^{(\text{train})\top}\boldsymbol{X}^{(\text{train})}\right)^{-1}\boldsymbol{X}^{(\text{train})\top}\boldsymbol{y}$$

The system of equations whose solution is given by equ the **normal equations**. Evaluating equation 5.12 constit algorithm. For an example of the linear regression learnir see figure 5.1.

It is worth noting that the term **linear regression** is a slightly more sophisticated model with one additional pa term b. In this model

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

so the mapping from parameters to predictions is still a l mapping from features to predictions is now an affine funct



affine functions means that the plot of the model's predic line, but it need not pass through the origin. Instead of add b, one can continue to use the model with only weights b extra entry that is always set to 1. The weight correspondi plays the role of the bias parameter. We frequently use t referring to affine functions throughout this book.

The intercept term b is often called the **bias** paramete mation. This terminology derives from the point of view transformation is biased toward being b in the absence of is different from the idea of a statistical bias, in which a algorithm's expected estimate of a quantity is not equal to

Linear regression is of course an extremely simple and lim but it provides an example of how a learning algorithm ca sections we describe some of the basic principles underlyi design and demonstrate how these principles can be used to l learning algorithms.

5.2 Capacity, Overfitting and Underfitt

The central challenge in machine learning is that our alg well on *new, previously unseen*inputs—not just those on trained. The ability to perform well on previously unobs **generalization**.

Typically, when training a machine learning model, we h set; we can compute some error measure on the training se **error**; and we reduce this training error. So far, what we h but we actually care about the test error, $\frac{1}{m^{(\text{test})}}||\boldsymbol{X}^{(\text{test})}\boldsymbol{w}$

How can we affect performance on the test set when w training set? The field of **statistical learning theory** prc the training and the test set are collected arbitrarily, there do. If we are allowed to make some assumptions about hov set are collected, then we can make some progress.

The training and test data are generated by a probab datasets called the **data-generating process**. We typic sumptions known collectively as the **i.i.d. assumptions**. that the examples in each dataset are **independent** from the training set and test set are **identically distributed**, probability distribution as each other. This assumption the data-generating process with a probability distribution The same distribution is then used to generate every train example. We call that shared underlying distribution the **tribution**, denoted p_{data}. This probabilistic framework anc enables us to mathematically study the relationship betw test error.

One immediate connection we can observe between train is that the expected training error of a randomly selected expected test error of that model. Suppose we have a p $p(\boldsymbol{x}, y)$ and we sample from it repeatedly to generate the tr set. For some fixed value $\boldsymbol{w}$, the expected training set error the expected test set error, because both expectations are dataset sampling process. The only difference between the name we assign to the dataset we sample.

Of course, when we use a machine learning algorith

obtain a sufficiently low error value on the training set. O
the gap between the training error and test error is too lar

We can control whether a model is more likely to overfit
its **capacity**. Informally, a model's capacity is its ability t
functions. Models with low capacity may struggle to fit th
with high capacity can overfit by memorizing properties of t
not serve them well on the test set.

One way to control the capacity of a learning algorit
hypothesis space, the set of functions that the learning a
select as being the solution. For example, the linear regres
set of all linear functions of its input as its hypothesis spa
linear regression to include polynomials, rather than just
hypothesis space. Doing so increases the model's capacity.

A polynomial of degree 1 gives us the linear regression
are already familiar, with the prediction

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

By introducing x^2 as another feature provided to the linea
can learn a model that is quadratic as a function of x:

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

Though this model implements a quadratic function of it
still a linear function of the *parameters*, so we can still use
to train the model in closed form. We can continue to ad
additional features, for example, to obtain a polynomial of

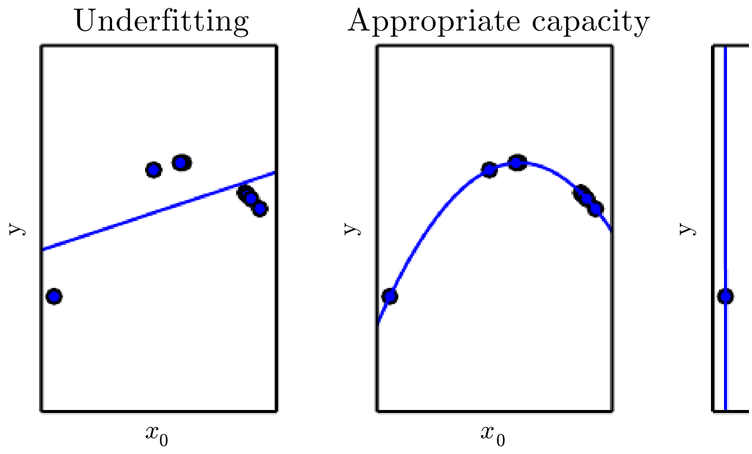


Figure 5.2: We fit three models to this example training set. generated synthetically, by randomly sampling x values and cho by evaluating a quadratic function. *(Left)*A linear function fit underfitting—it cannot capture the curvature that is present quadratic function fit to the data generalizes well to unseen point a significant amount of overfitting or underfitting. *(Right)*A p to the data suffers from overfitting. Here we used the Moore-F solve the underdetermined normal equations. The solution passe points exactly, but we have not been lucky enough for it to extr It now has a deep valley between two training points that doe underlying function. It also increases sharply on the left side of function decreases in this area.

function is quadratic. The linear function is unable to ca the true underlying problem, so it underfits. The degree-9 representing the correct function, but it is also capable of function within this family is a difficult optimization pro learning algorithm does not actually find the best functio significantly reduces the training error. These additional li imperfection of the optimization algorithm, mean that th **effective capacity** may be less than the representational family.

Our modern ideas about improving the generalizatio models are refinements of thought dating back to philosoph Ptolemy. Many early scholars invoke a principle of parsin widely known as **Occam's razor** (c. 1287–1347). This prin competing hypotheses that explain known observations e choose the "simplest" one. This idea was formalized and the twentieth century by the founders of statistical learnin Chervonenkis, 1971; Vapnik, 1982; Blumer *et al.*, 1989; Va

Statistical learning theory provides various means of qua Among these, the most well known is the **Vapnik-Chervo** VC dimension. The VC dimension measures the capacity of VC dimension is defined as being the largest possible valu exists a training set of m different $\boldsymbol{x}$ points that the classifi

Quantifying the capacity of the model enables statisti make quantitative predictions. The most important result theory show that the discrepancy between training error a is bounded from above by a quantity that grows as the mo shrinks as the number of training examples increases (Vap 1971; Vapnik, 1982; Blumer *et al.*, 1989; Vapnik, 1995). intellectual justification that machine learning algorithms rarely used in practice when working with deep learning

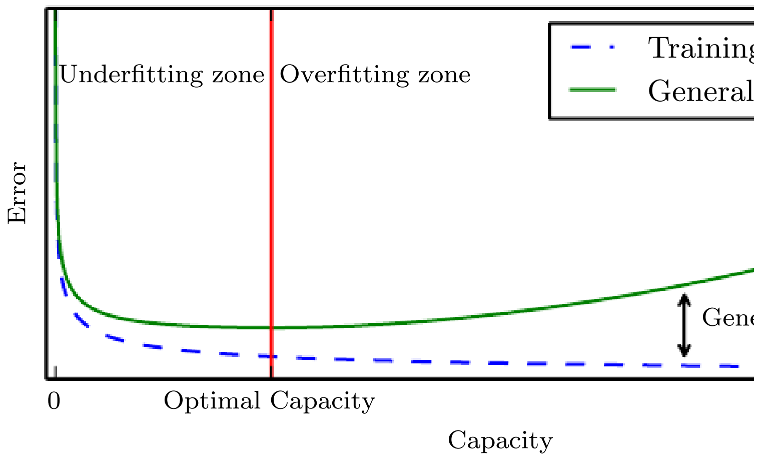


Figure 5.3: Typical relationship between capacity and error.
behave differently. At the left end of the graph, training error
are both high. This is the **underfitting regime**. As we increas
decreases, but the gap between training and generalization err
the size of this gap outweighs the decrease in training error, and
regime, where capacity is too large, above the **optimal capac**

generalization error has a U-shaped curve as a function of
illustrated in figure 5.3.

To reach the most extreme case of arbitrarily high c
the concept of **nonparametric** *models*. So far, we have
models, such as linear regression. Parametric models lear
by a parameter vector whose size is finite and fixed before
Nonparametric models have no such limitation.

Sometimes, nonparametric models are just theoretical might be greater than zero, if two identical inputs are ass outputs) on any regression dataset.

Finally, we can also create a nonparametric learning alg parametric learning algorithm inside another algorithm tha of parameters as needed. For example, we could imagine an that changes the degree of the polynomial learned by linear polynomial expansion of the input.

The ideal model is an oracle that simply knows the true p that generates the data. Even such a model will still incu problems, because there may still be some noise in the dis of supervised learning, the mapping from $\boldsymbol{x}$ to y may be or y may be a deterministic function that involves other v included in $\boldsymbol{x}$. The error incurred by an oracle making pre distribution $p(\boldsymbol{x}, y)$ is called the **Bayes error**.

Training and generalization error vary as the size of t Expected generalization error can never increase as the numb increases. For nonparametric models, more data yield bett the best possible error is achieved. Any fixed parametric optimal capacity will asymptote to an error value that ex See figure 5.4 for an illustration. Note that it is possible optimal capacity and yet still have a large gap between trai errors. In this situation, we may be able to reduce this g training examples.

5.2.1 The No Free Lunch Theorem

Learning theory claims that a machine learning algorithm c

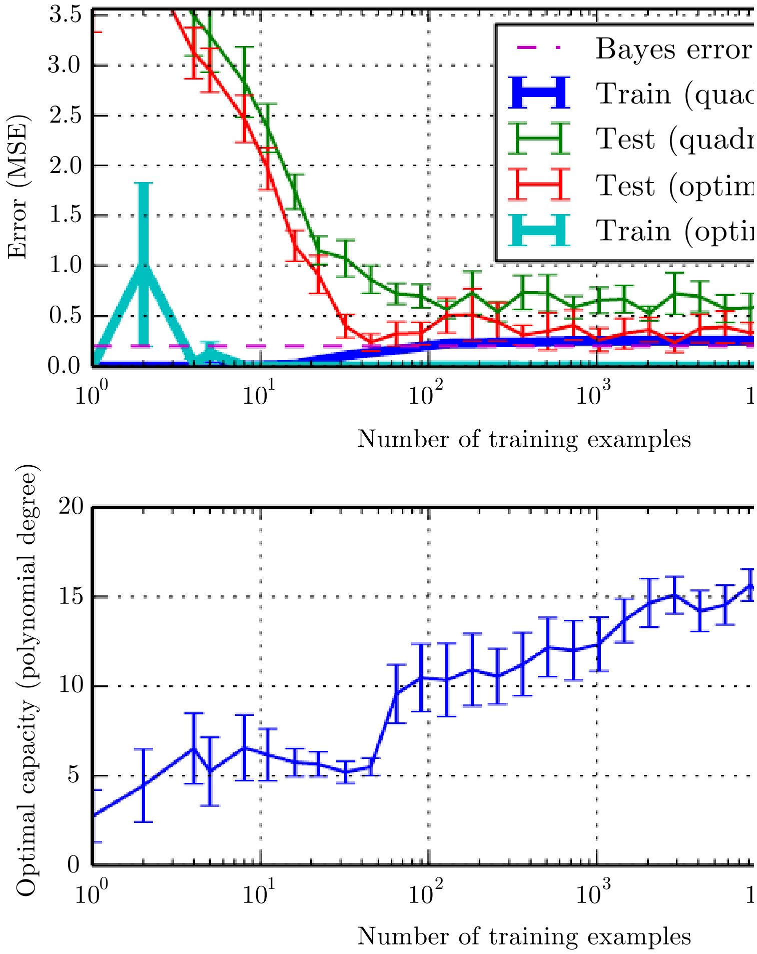


Figure 5.4: The effect of the training dataset size on the train a on the optimal model capacity. We constructed a synthetic regr adding a moderate amount of noise to a degree-5 polynomial, ge and then generated several different sizes of training set. For ea different training sets in order to plot error bars showing 95 per

same error rate when classifying previously unobserved p
in some sense, no machine learning algorithm is universall
other. The most sophisticated algorithm we can conceive o
performance (over all possible tasks) as merely predicting tl
to the same class.

Fortunately, these results hold only when we average
generating distributions. If we make assumptions about tl
distributions we encounter in real-world applications, then
algorithms that perform well on these distributions.

This means that the goal of machine learning research is
learning algorithm or the absolute best learning algorithm.
understand what kinds of distributions are relevant to the '
agent experiences, and what kinds of machine learning algo
data drawn from the kinds of data-generating distributions

5.2.2 Regularization

The no free lunch theorem implies that we must design
algorithms to perform well on a specific task. We do sc
preferences into the learning algorithm. When these prefer
the learning problems that we ask the algorithm to solve, i

So far, the only method of modifying a learning algorithm
concretely is to increase or decrease the model's representatic
or removing functions from the hypothesis space of solutions
is able to choose from. We gave the specific example of in
the degree of a polynomial for a regression problem. The v
so far is oversimplified.

in its hypothesis space. This means that both functions a
preferred. The unpreferred solution will be chosen only if i
significantly better than the preferred solution.

For example, we can modify the training criterion for line
weight decay. To perform linear regression with weight dec
comprising both the mean squared error on the training anc
expresses a preference for the weights to have smaller squarec

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

where λ is a value chosen ahead of time that controls the str
for smaller weights. When $\lambda = 0$, we impose no preference,
weights to become smaller. Minimizing $J(\boldsymbol{w})$ results in a
make a tradeoff between fitting the training data and beir
solutions that have a smaller slope, or that put weight or
As an example of how we can control a model's tendency
via weight decay, we can train a high-degree polynomial
different values of λ. See figure 5.5 for the results.

More generally, we can regularize a model that learns
adding a penalty called a **regularizer** to the cost function
decay, the regularizer is $\Omega(\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{w}$. In chapter 7, we w
regularizers are possible.

Expressing preferences for one function over another
of controlling a model's capacity than including or exclud
hypothesis space. We can think of excluding a function fron
expressing an infinitely strong preference against that func

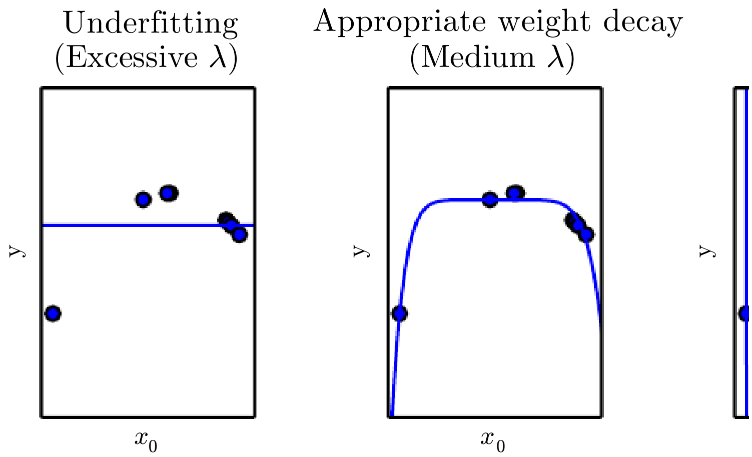
In our weight decay example, we expressed our preferer

people can do) may all be solved effectively using very ge
regularization.

5.3 Hyperparameters and Validation S

Most machine learning algorithms have hyperparameters
use to control the algorithm's behavior. The values of hy
adapted by the learning algorithm itself (though we can de
procedure in which one learning algorithm learns the bes
another learning algorithm).

The polynomial regression example in figure 5.2 has a
the degree of the polynomial, which acts as a **capacity**
λ value used to control the strength of weight decay is
hyperparameter.



Sometimes a setting is chosen to be a hyperparameter
rithm does not learn because the setting is difficult to opti
the setting must be a hyperparameter because it is not ap
hyperparameter on the training set. This applies to all
control model capacity. If learned on the training set, such h
always choose the maximum possible model capacity, result
to figure 5.3). For example, we can always fit the train
higher-degree polynomial and a weight decay setting of λ =
a lower-degree polynomial and a positive weight decay sett

To solve this problem, we need a **validation set** of exa
algorithm does not observe.

Earlier we discussed how a held-out test set, composed of
the same distribution as the training set, can be used to esti
error of a learner, after the learning process has completed. I
test examples are not used in any way to make choices abou
its hyperparameters. For this reason, no example from th
in the validation set. Therefore, we always construct the v
training data. Specifically, we split the training data int
One of these subsets is used to learn the parameters. T
validation set, used to estimate the generalization error d
allowing for the hyperparameters to be updated according
used to learn the parameters is still typically called the tra
this may be confused with the larger pool of data used
process. The subset of data used to guide the selection
called the validation set. Typically, one uses about 80 p
data for training and 20 percent for validation. Since the
to “train” the hyperparameters, the validation set error v

5.3.1 Cross-Validation

Dividing the dataset into a fixed training set and a fixed test if it results in the test set being small. A small test set implie around the estimated average test error, making it difficult t A works better than algorithm B on the given task.

When the dataset has hundreds of thousands of example serious issue. When the dataset is too small, are alternative to use all the examples in the estimation of the mean tes increased computational cost. These procedures are based o the training and testing computation on different randomly of the original dataset. The most common of these is the procedure, shown in algorithm 5.1, in which a partition of t splitting it into k nonoverlapping subsets. The test error by taking the average test error across k trials. On trial i, data is used as the test set, and the rest of the data is us One problem is that no unbiased estimators of the varianc estimators exist (Bengio and Grandvalet, 2004), but appro used.

5.4 Estimators, Bias and Variance

The field of statistics gives us many tools to achieve the m solving a task not only on the training set but also to ge concepts such as parameter estimation, bias and variance characterize notions of generalization, underfitting and ove

Algorithm 5.1 The k-fold cross-validation algorithm. It c
generalization error of a learning algorithm A when the
small for a simple train/test or train/valid split to yield a
generalization error, because the mean of a loss L on a smal
high a variance. The dataset $\mathbb{D}$ contains as elements the abs
the i-th example), which could stand for an (input,target
in the case of supervised learning, or for just an input
of unsupervised learning. The algorithm returns the vect
example in $\mathbb{D}$, whose mean is the estimated generalizatio
individual examples can be used to compute a confidenc
mean (equation 5.47). Though these confidence intervals
after the use of cross-validation, it is still common practice
that algorithm A is better than algorithm B only if the co
error of algorithm A lies below and does not intersect the
algorithm B.

```
Define KFoldXV($\mathbb{D}, A, L, k$):
Require: $\mathbb{D}$, the given dataset, with elements $\boldsymbol{z}^{(i)}$
Require: $A$, the learning algorithm, seen as a function t
   input and outputs a learned function
Require: $L$, the loss function, seen as a function from a l
   an example $\boldsymbol{z}^{(i)} \in \mathbb{D}$ to a scalar $\in \mathbb{R}$
Require: $k$, the number of folds
   Split $\mathbb{D}$ into $k$ mutually exclusive subsets $\mathbb{D}_i$, whose unio
   for $i$ from 1 to $k$ do
      $f_i = A(\mathbb{D} \backslash \mathbb{D}_i)$
      for $\boldsymbol{z}^{(j)}$ in $\mathbb{D}_i$ do
         $e_j = L(f_i, \boldsymbol{z}^{(j)})$
      end for
```

a good estimator is a function whose output is close to the
generated the training data.

For now, we take the frequentist perspective on statisti
that the true parameter value $\boldsymbol{\theta}$ is fixed but unknown, wh
$\hat{\boldsymbol{\theta}}$ is a function of the data. Since the data is drawn from a
function of the data is random. Therefore $\hat{\boldsymbol{\theta}}$ is a random va

Point estimation can also refer to the estimation of th
input and target variables. We refer to these types of poin
estimators.

Function Estimation Sometimes we are interested in
estimation (or function approximation). Here, we are tryin
$\boldsymbol{y}$ given an input vector $\boldsymbol{x}$. We assume that there is a functi
the approximate relationship between $\boldsymbol{y}$ and $\boldsymbol{x}$. For exa
that $\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ stands for the part of $\boldsymbol{y}$ that i
$\boldsymbol{x}$. In function estimation, we are interested in approximat
estimate $\hat{f}$. Function estimation is really just the same as e
$\boldsymbol{\theta}$; the function estimator $\hat{f}$ is simply a point estimator in
linear regression example (discussed in section 5.1.4) and the
example (discussed in section 5.2) both illustrate scenarios t
as either estimating a parameter $\boldsymbol{w}$ or estimating a functi
to y.

We now review the most commonly studied properties o
discuss what they tell us about these estimators.

5.4.2 Bias

bution with mean θ:

$$P(x^{(i)};\theta) = \theta^{x^{(i)}}(1-\theta)^{(1-x^{(i)})}.$$

A common estimator for the θ parameter of this distribut
training samples:

$$\hat{\theta}_m = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}.$$

To determine whether this estimator is biased, we can su
into equation 5.20:

$$\begin{aligned}
\text{bias}(\hat{\theta}_m) &= \mathbb{E}[\hat{\theta}_m] - \theta \\
&= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right] - \theta \\
&= \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\left[x^{(i)}\right] - \theta \\
&= \frac{1}{m}\sum_{i=1}^{m} \sum_{x^{(i)}=0}^{1} \Big(x^{(i)}\theta^{x^{(i)}}(1-\theta)^{(1-x^{(}} \\
&= \frac{1}{m}\sum_{i=1}^{m} (\theta) - \theta \\
&= \theta - \theta = 0
\end{aligned}$$

Since $\text{bias}(\hat{\theta}) = 0$, we say that our estimator $\hat{\theta}$ is unbias

Example: Gaussian Distribution Estimator of the I

To determine the bias of the sample mean, we are again ir
its expectation:

$$\begin{aligned}\text{bias}(\hat{\mu}_m) &= \mathbb{E}[\hat{\mu}_m] - \mu \\ &= \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right] - \mu \\ &= \left(\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}\left[x^{(i)}\right]\right) - \mu \\ &= \left(\frac{1}{m}\sum_{i=1}^{m} \mu\right) - \mu \\ &= \mu - \mu = 0\end{aligned}$$

Thus we find that the sample mean is an unbiased estima
parameter.

Example: Estimators of the Variance of a Gaussia this example, we compare two different estimators of the va
a Gaussian distribution. We are interested in knowing if eit

The first estimator of σ^2 we consider is known as the **s**

$$\hat{\sigma}_m^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \hat{\mu}_m\right)^2,$$

where $\hat{\mu}_m$ is the sample mean. More formally, we are intere

The **unbiased sample variance** estimator

$$\tilde{\sigma}_m^2 = \frac{1}{m-1}\sum_{i=1}^{m}\left(x^{(i)} - \hat{\mu}_m\right)^2$$

provides an alternative approach. As the name suggests thi
That is, we find that $\mathbb{E}[\tilde{\sigma}_m^2] = \sigma^2$:

$$\begin{aligned}\mathbb{E}[\tilde{\sigma}_m^2] &= \mathbb{E}\left[\frac{1}{m-1}\sum_{i=1}^{m}\left(x^{(i)} - \hat{\mu}_m\right)^2\right] \\ &= \frac{m}{m-1}\mathbb{E}[\hat{\sigma}_m^2] \\ &= \frac{m}{m-1}\left(\frac{m-1}{m}\sigma^2\right) \\ &= \sigma^2.\end{aligned}$$

We have two estimators: one is biased, and the other i
estimators are clearly desirable, they are not always the "b
will see we often use biased estimators that possess other i

5.4.3 Variance and Standard Error

Another property of the estimator that we might want to
we expect it to vary as a function of the data sample. Jus
expectation of the estimator to determine its bias, we can
The **variance** of an estimator is simply the variance

been different. The expected degree of variation in any es
error that we want to quantify.

The standard error of the mean is given by

$$\mathrm{SE}(\hat{\mu}_m) = \sqrt{\mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} x^{(i)}\right]} = \frac{\sigma}{\sqrt{m}}$$

where σ^2 is the true variance of the samples x^i. The st
estimated by using an estimate of σ. Unfortunately, neit
the sample variance nor the square root of the unbiased est
provide an unbiased estimate of the standard deviation.
to underestimate the true standard deviation but are still
square root of the unbiased estimator of the variance is les
For large m, the approximation is quite reasonable.

The standard error of the mean is very useful in machin
We often estimate the generalization error by computing t
error on the test set. The number of examples in the te
accuracy of this estimate. Taking advantage of the centra
tells us that the mean will be approximately distributed wit
we can use the standard error to compute the probability th
falls in any chosen interval. For example, the 95 percent confi
on the mean $\hat{\mu}_m$ is

$$(\hat{\mu}_m - 1.96\mathrm{SE}(\hat{\mu}_m), \hat{\mu}_m + 1.96\mathrm{SE}(\hat{\mu}_m))$$

under the normal distribution with mean $\hat{\mu}_m$ and variance
learning experiments, it is common to say that algorithm A i

$$
\begin{aligned}
&= \frac{1}{m^2}\sum_{i=1}^{m} \mathrm{Var}\left(x^{(i)}\right) \\
&= \frac{1}{m^2}\sum_{i=1}^{m} \theta(1-\theta) \\
&= \frac{1}{m^2} m\theta(1-\theta) \\
&= \frac{1}{m}\theta(1-\theta)
\end{aligned}
$$

The variance of the estimator decreases as a function of m, t in the dataset. This is a common property of popular es return to when we discuss consistency (see section 5.4.5).

5.4.4 Trading off Bias and Variance to Minim Error

Bias and variance measure two different sources of error measures the expected deviation from the true value of the Variance on the other hand, provides a measure of the devia estimator value that any particular sampling of the data is

What happens when we are given a choice between tw more bias and one with more variance? How do we choos example, imagine that we are interested in approximating figure 5.2 and we are only offered the choice between a mo one that suffers from large variance. How do we choose be

The most common way to negotiate this trade-off is t Empirically, cross-validation is highly successful on many r

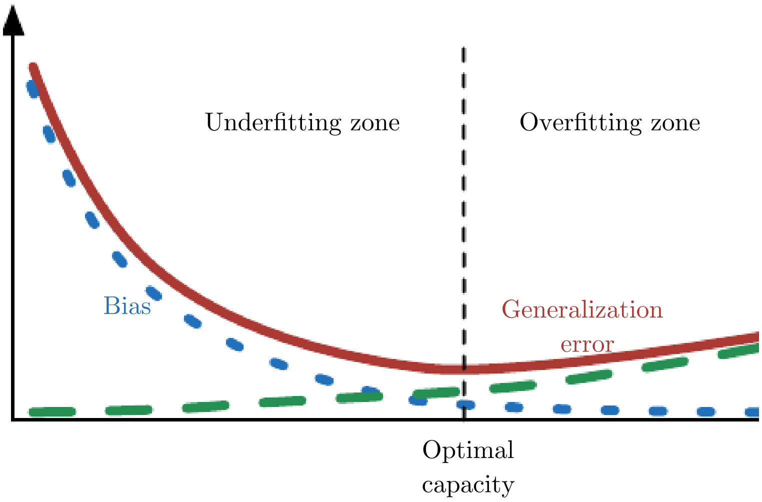


Figure 5.6: As capacity increases (x-axis), bias (dotted) tends t
(dashed) tends to increase, yielding another U-shaped curve for g
curve). If we vary capacity along one axis, there is an optimal ca
when the capacity is below this optimum and overfitting when it is
is similar to the relationship between capacity, underfitting, and
section 5.2 and figure 5.3.

error is measured by the MSE (where bias and variance are r
of generalization error), increasing capacity tends to increase
bias. This is illustrated in figure 5.6, where we see again
generalization error as a function of capacity.

5.4.5 Consistency

So far we have discussed the properties of various estimato

sure convergence of a sequence of random variables $\mathbf{x}^{(1}$
occurs when $p(\lim_{m\to\infty} \mathbf{x}^{(m)} = \boldsymbol{x}) = 1$.

Consistency ensures that the bias induced by the estim
number of data examples grows. However, the reverse is
unbiasedness does not imply consistency. For example, c
mean parameter μ of a normal distribution $\mathcal{N}(x; \mu, \sigma^2)$, wi
of m samples: $\{x^{(1)}, \ldots, x^{(m)}\}$. We could use the first sam
as an unbiased estimator: $\hat{\theta} = x^{(1)}$. In that case, $\mathbb{E}(\hat{\theta}_m)$
is unbiased no matter how many data points are seen. T
that the estimate is asymptotically unbiased. However, t
estimator as it is *not* the case that $\hat{\theta}_m \to \theta$ as $m \to \infty$.

5.5 Maximum Likelihood Estimation

We have seen some definitions of common estimators and an
But where did these estimators come from? Rather tha
function might make a good estimator and then analyzing
we would like to have some principle from which we can d
that are good estimators for different models.

The most common such principle is the maximum likel

Consider a set of m examples $\mathbb{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ dra
the true but unknown data-generating distribution $p_{\text{data}}(\mathbf{x}$

Let $p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$ be a parametric family of probability
same space indexed by $\boldsymbol{\theta}$. In other words, $p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})$ ma
to a real number estimating the true probability $p_{\text{data}}(\boldsymbol{x})$.

The maximum likelihood estimator for $\boldsymbol{\theta}$ is then define into a sum:

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\mathrm{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

Because the $\arg\max$ does not change when we rescale the divide by m to obtain a version of the criterion that is expr with respect to the empirical distribution $\hat{p}_{\mathrm{data}}$ defined by

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathrm{data}}} \log p_{\mathrm{model}}(\boldsymbol{x}; \boldsymbol{\theta}$$

One way to interpret maximum likelihood estimation is t the dissimilarity between the empirical distribution $\hat{p}_{\mathrm{data}}$, set and the model distribution, with the degree of dissimil measured by the KL divergence. The KL divergence is giv

$$D_{\mathrm{KL}}\left(\hat{p}_{\mathrm{data}} \| p_{\mathrm{model}}\right) = \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathrm{data}}}\left[\log \hat{p}_{\mathrm{data}}(\boldsymbol{x}) - \log p_{\mathrm{m}}\right.$$

The term on the left is a function only of the data-gener model. This means when we train the model to minimize need only minimize

$$-\mathbb{E}_{\mathbf{x}\sim\hat{p}_{\mathrm{data}}}\left[\log p_{\mathrm{model}}(\boldsymbol{x})\right],$$

which is of course the same as the maximization in equatio

Minimizing this KL divergence corresponds exactly to entropy between the distributions. Many authors use the t identify specifically the negative log-likelihood of a Bernoulli but that is a misnomer. Any loss consisting of a negative l entropy between the empirical distribution defined by th probability distribution defined by model. For example, me cross-entropy between the empirical distribution and a Ga

5.5.1 Conditional Log-Likelihood and Mean S

The maximum likelihood estimator can readily be generaliz tional probability $P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})$ in order to predict $\mathbf{y}$ given most common situation because it forms the basis for most $\boldsymbol{X}$ represents all our inputs and $\boldsymbol{Y}$ all our observed target maximum likelihood estimator is

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{Y} \mid \boldsymbol{X}; \boldsymbol{\theta}).$$

If the examples are assumed to be i.i.d., then this can be

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log P(\boldsymbol{y}^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}$$

Example: Linear Regression as Maximum Likeliho introduced in section 5.1.4, may be justified as a maximur Previously, we motivated linear regression as an algorithm input $\boldsymbol{x}$ and produce an output value $\hat{y}$. The mapping fr minimize mean squared error, a criterion that we introduced We now revisit linear regression from the point of view o estimation. Instead of producing a single prediction $\hat{y}$, we as producing a conditional distribution $p(y \mid \boldsymbol{x})$. We can infinitely large training set, we might see several training e input value $\boldsymbol{x}$ but different values of y. The goal of the lea to fit the distribution $p(y \mid \boldsymbol{x})$ to all those different y values with $\boldsymbol{x}$. To derive the same linear regression algorithm w

where $\hat{y}^{(i)}$ is the output of the linear regression on the i-th i
number of the training examples. Comparing the log-like
squared error,

$$\mathrm{MSE}_{\mathrm{train}} = \frac{1}{m}\sum_{i=1}^{m} ||\hat{y}^{(i)} - y^{(i)}||^2,$$

we immediately see that maximizing the log-likelihood wi
the same estimate of the parameters $\boldsymbol{w}$ as does minimizing t
The two criteria have different values but the same locatior
justifies the use of the MSE as a maximum likelihood estima
will see, the maximum likelihood estimator has several des

5.5.2 Properties of Maximum Likelihood

The main appeal of the maximum likelihood estimator is t
be the best estimator asymptotically, as the number of exan
of its rate of convergence as m increases.

Under appropriate conditions, the maximum likelihc
property of consistency (see section 5.4.5), meaning that as
examples approaches infinity, the maximum likelihood es
converges to the true value of the parameter. These condit

- The true distribution p_{data} must lie within the mo
Otherwise, no estimator can recover p_{data}.

- The true distribution p_{data} must correspond to exact
erwise, maximum likelihood can recover the correct p_{d}
to determine which value of $\boldsymbol{\theta}$ was used by the data-

values, where the expectation is over m training samples fro
distribution. That parametric mean squared error decreas
for m large, the Cramér-Rao lower bound (Rao, 1945; Cra
no consistent estimator has a lower MSE than the maximu

For these reasons (consistency and efficiency), maxim
considered the preferred estimator to use for machine learn
of examples is small enough to yield overfitting behavior, re
such as weight decay may be used to obtain a biased version
that has less variance when training data is limited.

5.6 Bayesian Statistics

So far we have discussed **frequentist statistics** and approa
ing a single value of $\boldsymbol{\theta}$, then making all predictions therea
estimate. Another approach is to consider all possible valu
prediction. The latter is the domain of **Bayesian statisti**

As discussed in section 5.4.1, the frequentist perspe
parameter value $\boldsymbol{\theta}$ is fixed but unknown, while the point e
variable on account of it being a function of the dataset (wh

The Bayesian perspective on statistics is quite differe
probability to reflect degrees of certainty in states of know
directly observed and so is not random. On the other hand
is unknown or uncertain and thus is represented as a rando

Before observing the data, we represent our knowledg
probability distribution, $p(\boldsymbol{\theta})$ (sometimes referred to
Generally, the machine learning practitioner selects a pri

In the scenarios where Bayesian estimation is typically usec relatively uniform or Gaussian distribution with high entrop of the data usually causes the posterior to lose entropy anc few highly likely values of the parameters.

Relative to maximum likelihood estimation, Bayesian important differences. First, unlike the maximum likelihoo predictions using a point estimate of $\boldsymbol{\theta}$, the Bayesian approac using a full distribution over $\boldsymbol{\theta}$. For example, after obser predicted distribution over the next data sample, $x^{(m+1)}$, i

$$p(x^{(m+1)} \mid x^{(1)}, \dots, x^{(m)}) = \int p(x^{(m+1)} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid x^{(1)}, \dots$$

Here each value of $\boldsymbol{\theta}$ with positive probability density contri of the next example, with the contribution weighted by the After having observed $\{x^{(1)}, \dots, x^{(m)}\}$, if we are still quit value of $\boldsymbol{\theta}$, then this uncertainty is incorporated directly ir might make.

In section 5.4, we discussed how the frequentist approa tainty in a given point estimate of $\boldsymbol{\theta}$ by evaluating its var the estimator is an assessment of how the estimate might c samplings of the observed data. The Bayesian answer to the with the uncertainty in the estimator is to simply integrate protect well against overfitting. This integral is of course the laws of probability, making the Bayesian approach simp frequentist machinery for constructing an estimator is base decision to summarize all knowledge contained in the data estimate.

Example: Bayesian Linear Regression Here we cons
mation approach to learning the linear regression paramete
we learn a linear mapping from an input vector $\boldsymbol{x} \in \mathbb{R}^n$ t
scalar $y \in \mathbb{R}$. The prediction is parametrized by the vector

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}.$$

Given a set of m training samples $(\boldsymbol{X}^{(\text{train})}, \boldsymbol{y}^{(\text{train})})$, we ca
of y over the entire training set as

$$\hat{\boldsymbol{y}}^{(\text{train})} = \boldsymbol{X}^{(\text{train})}\boldsymbol{w}.$$

Expressed as a Gaussian conditional distribution on $\boldsymbol{y}^{(}$

$$p(\boldsymbol{y}^{(\text{train})} \mid \boldsymbol{X}^{(\text{train})}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}^{(\text{train})}; \boldsymbol{X}^{(\text{train})}\boldsymbol{w}, \boldsymbol{I})$$

$$\propto \exp\left(-\frac{1}{2}(\boldsymbol{y}^{(\text{train})} - \boldsymbol{X}^{(\text{train})}\boldsymbol{w})^\top (\right.$$

where we follow the standard MSE formulation in assum
variance on y is one. In what follows, to reduce the notatio
$(\boldsymbol{X}^{(\text{train})}, \boldsymbol{y}^{(\text{train})})$ as simply $(\boldsymbol{X}, \boldsymbol{y})$.

To determine the posterior distribution over the model
first need to specify a prior distribution. The prior should
about the value of these parameters. While it is sometime
to express our prior beliefs in terms of the parameters of th
typically assume a fairly broad distribution, expressing a hig
about $\boldsymbol{\theta}$. For real-valued parameters it is common to use

$$\propto \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})\right)\exp\left(-\frac{1}{2}(\right.$$

$$\propto \exp\left(-\frac{1}{2}\left(-2\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{w}+\boldsymbol{w}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{w}+\boldsymbol{w}^\top\boldsymbol{\Lambda}\right.\right.$$

We now define $\boldsymbol{\Lambda}_m = \left(\boldsymbol{X}^\top\boldsymbol{X}+\boldsymbol{\Lambda}_0^{-1}\right)^{-1}$ and $\boldsymbol{\mu}_m = \boldsymbol{\Lambda}_m$
ing these new variables, we find that the posterior may be r
distribution:

$$\begin{aligned}p(\boldsymbol{w}\mid\boldsymbol{X},\boldsymbol{y}) &\propto \exp\left(-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{\mu}_m)^\top\boldsymbol{\Lambda}_m^{-1}(\boldsymbol{w}-\boldsymbol{\mu}_m)+\frac{1}{2}\right.\\ &\propto \exp\left(-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{\mu}_m)^\top\boldsymbol{\Lambda}_m^{-1}(\boldsymbol{w}-\boldsymbol{\mu}_m)\right).\end{aligned}$$

All terms that do not include the parameter vector $\boldsymbol{w}$ ha
are implied by the fact that the distribution must be norma
Equation 3.23 shows how to normalize a multivariate Gaus

Examining this posterior distribution enables us to gain
effect of Bayesian inference. In most situations, we set $\boldsymbol{\mu}_0$ tc
then μ_m gives the same estimate of $\boldsymbol{w}$ as does frequentist l
weight decay penalty of $\alpha\boldsymbol{w}^\top\boldsymbol{w}$. One difference is that th
undefined if α is set to zero—we are not allowed to begin
process with an infinitely wide prior on $\boldsymbol{w}$. The more impo
the Bayesian estimate provides a covariance matrix, show
different values of $\boldsymbol{w}$ are, rather than providing only the es

maximal posterior probability (or maximal probability densi
case of continuous $\boldsymbol{\theta}$):

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{x} \mid \boldsymbol{\theta}) +$$

We recognize, on the righthand side, $\log p(\boldsymbol{x} \mid \boldsymbol{\theta})$, tha
likelihood term, and $\log p(\boldsymbol{\theta})$, corresponding to the prior di

As an example, consider a linear regression model wit
the weights $\boldsymbol{w}$. If this prior is given by $\mathcal{N}(\boldsymbol{w}; \boldsymbol{0}, \frac{1}{\lambda}\boldsymbol{I}^2)$, the
equation 5.79 is proportional to the familiar $\lambda \boldsymbol{w}^\top \boldsymbol{w}$ weigh
term that does not depend on $\boldsymbol{w}$ and does not affect the l
Bayesian inference with a Gaussian prior on the weights thu
decay.

As with full Bayesian inference, MAP Bayesian inferenc
leveraging information that is brought by the prior and c
training data. This additional information helps to redu
MAP point estimate (in comparison to the ML estimate).
the price of increased bias.

Many regularized estimation strategies, such as maxim
regularized with weight decay, can be interpreted as makin
tion to Bayesian inference. This view applies when the reg
adding an extra term to the objective function that corres
all regularization penalties correspond to MAP Bayesian i
some regularizer terms may not be the logarithm of a p
Other regularization terms depend on the data, which of co
distribution is not allowed to do.

“supervisor,” but the term still applies even when the tra
collected automatically.

5.7.1 Probabilistic Supervised Learning

Most supervised learning algorithms in this book are b
probability distribution $p(y \mid \boldsymbol{x})$. We can do this simpl
likelihood estimation to find the best parameter vector $\boldsymbol{\theta}$ f
of distributions $p(y \mid \boldsymbol{x}; \boldsymbol{\theta})$.

We have already seen that linear regression correspond

$$p(y \mid \boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{N}(y; \boldsymbol{\theta}^\top \boldsymbol{x}, \boldsymbol{I}).$$

We can generalize linear regression to the classification s
different family of probability distributions. If we have t
class 1, then we need only specify the probability of one
probability of class 1 determines the probability of class 0, b
must add up to 1.

The normal distribution over real-valued numbers th
regression is parametrized in terms of a mean. Any value w
is valid. A distribution over a binary variable is slightly mor
its mean must always be between 0 and 1. One way to solv
the logistic sigmoid function to squash the output of the l
interval (0, 1) and interpret that value as a probability:

$$p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}).$$

This approach is known as **logistic regression** (a somewh

5.7.2 Support Vector Machines

One of the most influential approaches to supervised learnin machine (Boser *et al.*, 1992; Cortes and Vapnik, 1995). T logistic regression in that it is driven by a linear function $\boldsymbol{w}$ regression, the support vector machine does not provide outputs a class identity. The SVM predicts that the positiv $\boldsymbol{w}^\top \boldsymbol{x} + b$ is positive. Likewise, it predicts that the negativ $\boldsymbol{w}^\top \boldsymbol{x} + b$ is negative.

One key innovation associated with support vector m **trick**. The kernel trick consists of observing that many mach can be written exclusively in terms of dot products between it can be shown that the linear function used by the suppo be re-written as

$$\boldsymbol{w}^\top \boldsymbol{x} + b = b + \sum_{i=1}^{m} \alpha_i \boldsymbol{x}^\top \boldsymbol{x}^{(i)},$$

where $\boldsymbol{x}^{(i)}$ is a training example, and $\boldsymbol{\alpha}$ is a vector of coeffi learning algorithm this way enables us to replace $\boldsymbol{x}$ with the o function $\phi(\boldsymbol{x})$ and the dot product with a function $k(\boldsymbol{x}, \boldsymbol{x}^{(i)}$ a **kernel**. The $\cdot$ operator represents an inner product anal For some feature spaces, we may not use literally the vec some infinite dimensional spaces, we need to use other kind example, inner products based on integration rather than su development of these kinds of inner products is beyond the

After replacing dot products with kernel evaluations, we using the function

$$f(\boldsymbol{x}) = b + \sum \alpha_i k(\boldsymbol{x}, \boldsymbol{x}^{(i)})$$

admits an implementation that is significantly more comput
naively constructing two $\phi(\boldsymbol{x})$ vectors and explicitly taking

In some cases, $\phi(\boldsymbol{x})$ can even be infinite dimensional,
an infinite computational cost for the naive, explicit app
$k(\boldsymbol{x}, \boldsymbol{x}')$ is a nonlinear, tractable function of $\boldsymbol{x}$ even when
an example of an infinite-dimensional feature space with
construct a feature mapping $\phi(x)$ over the nonnegative in
this mapping returns a vector containing x ones followed b
We can write a kernel function $k(x, x^{(i)}) = \min(x, x^{(i)})$ tha
to the corresponding infinite-dimensional dot product.

The most commonly used kernel is the **Gaussian kern**

$$k(\boldsymbol{u}, \boldsymbol{v}) = \mathcal{N}(\boldsymbol{u} - \boldsymbol{v}; 0, \sigma^2 \boldsymbol{I}),$$

where $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the standard normal density. This k
the **radial basis function** (RBF) kernel, because its valu
in $\boldsymbol{v}$ space radiating outward from $\boldsymbol{u}$. The Gaussian kerne
product in an infinite-dimensional space, but the derivati
straightforward than in our example of the min kernel ove

We can think of the Gaussian kernel as performing a kin
ing. A training example $\boldsymbol{x}$ associated with training label
for class y. When a test point $\boldsymbol{x}'$ is near $\boldsymbol{x}$ according to E
Gaussian kernel has a large response, indicating that $\boldsymbol{x}'$ i
template. The model then puts a large weight on the asso
Overall, the prediction will combine many such training l
similarity of the corresponding training examples.

Support vector machines are not the only algorithm
using the kernel trick. Many other linear models can be enh

generic kernels struggle to generalize well. We explain wh modern incarnation of deep learning was designed to overco kernel machines. The current deep learning renaissance be; (2006) demonstrated that a neural network could outperforn on the MNIST benchmark.

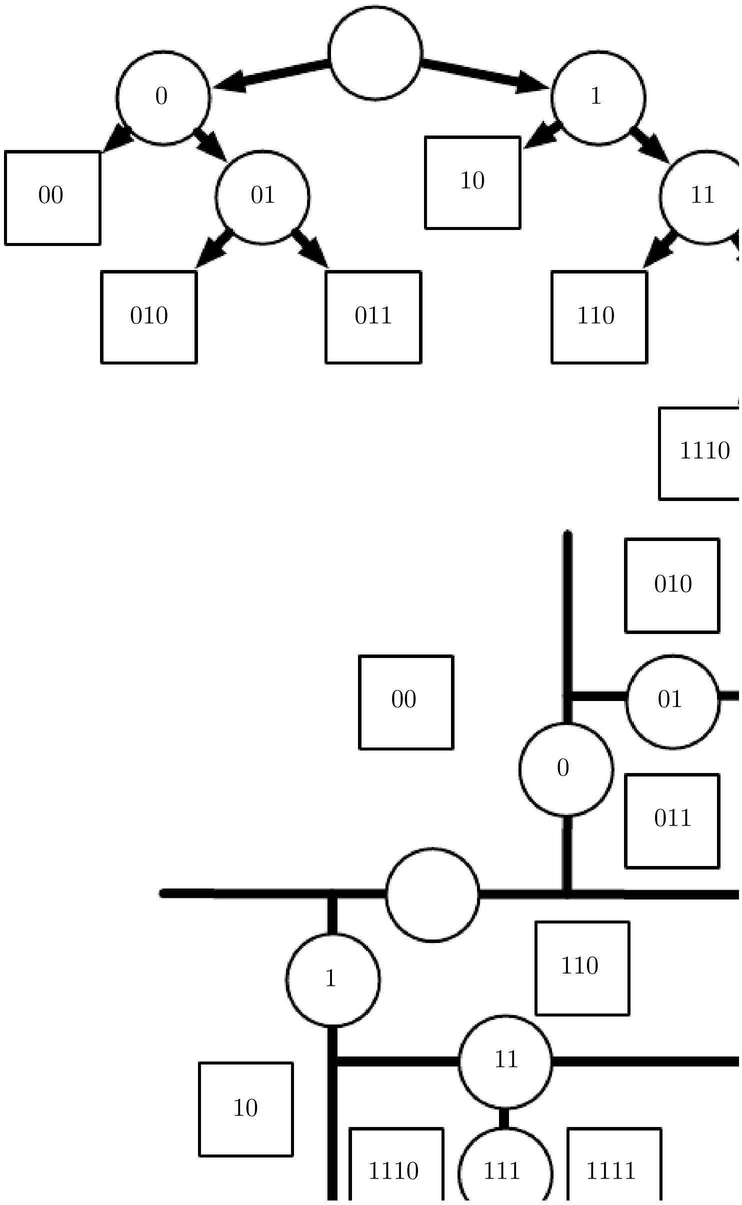
5.7.3 Other Simple Supervised Learning Algoı

We have already briefly encountered another nonprobabilis algorithm, nearest neighbor regression. More generally, a family of techniques that can be used for classificatioı nonparametric learning algorithm, k-nearest neighbors is n number of parameters. We usually think of the k-neares as not having any parameters but rather implementing a training data. In fact, there is not even really a training sta Instead, at test time, when we want to produce an output y we find the k-nearest neighbors to $\boldsymbol{x}$ in the training data $\mathbb{X}$ average of the corresponding y values in the training set. Tl any kind of supervised learning where we can define an ave the case of classification, we can average over one-hot code and $c_i = 0$ for all other values of i. We can then interpret one-hot codes as giving a probability distribution over classe learning algorithm, k-nearest neighbor can achieve very high suppose we have a multiclass classification task and measur loss. In this setting, 1-nearest neighbor converges to double number of training examples approaches infinity. The error error results from choosing a single neighbor by breakinş distant neighbors randomly. When there is infinite training

The nearest neighbor of most points $\boldsymbol{x}$ will be determined l
features x_2 through x_{100}, not by the lone feature x_1. Thu
training sets will essentially be random.

Another type of learning algorithm that also breaks the i
and has separate parameters for each region is the **decisio**
1984) and its many variants. As shown in figure 5.7, eacl
tree is associated with a region in the input space, and int
region into one subregion for each child of the node (typicall
cut). Space is thus subdivided into nonoverlapping regic
correspondence between leaf nodes and input regions. Each
every point in its input region to the same output. Dec
trained with specialized algorithms that are beyond the sc
learning algorithm can be considered nonparametric if it is
of arbitrary size, though decision trees are usually regularize
that turn them into parametric models in practice. Deci
typically used, with axis-aligned splits and constant outp
struggle to solve some problems that are easy even for lo
example, if we have a two-class problem, and the positive
$x_2 > x_1$, the decision boundary is not axis aligned. The
need to approximate the decision boundary with many node
function that constantly walks back and forth across the
with axis-aligned steps.

As we have seen, nearest neighbor predictors and dec
limitations. Nonetheless, they are useful learning algorithm
resources are constrained. We can also build intuition f
learning algorithms by thinking about the similarities ar
sophisticated algorithms and k-nearest neighbors or decisic


0
1
00
01
10
11
010
011
110
1110
010
00
01
0
011
110
1
11
10
1110
111
1111

examples. The term is usually associated with density e
draw samples from a distribution, learning to denoise data f
finding a manifold that the data lies near, or clustering th
related examples.

A classic unsupervised learning task is to find the "best
data. By "best" we can mean different things, but generally s
for a representation that preserves as much information abo
obeying some penalty or constraint aimed at keeping the rep
more accessible than $\boldsymbol{x}$ itself.

There are multiple ways of defining a simpler represe
most common include lower-dimensional representations, s
and independent representations. Low-dimensional repre
compress as much information about x as possible in a s
Sparse representations (Barlow, 1989; Olshausen and Fi
Ghahramani, 1997) embed the dataset into a representat
mostly zeros for most inputs. The use of sparse representat
increasing the dimensionality of the representation, so th
becoming mostly zeros does not discard too much informat
overall structure of the representation that tends to distribu
of the representation space. Independent representations a
the sources of variation underlying the data distribution su
of the representation are statistically independent.

Of course these three criteria are certainly not mut
dimensional representations often yield elements that hav
pendencies than the original high-dimensional data. This
reduce the size of a representation is to find and remove red
and removing more redundancy enables the dimensionality

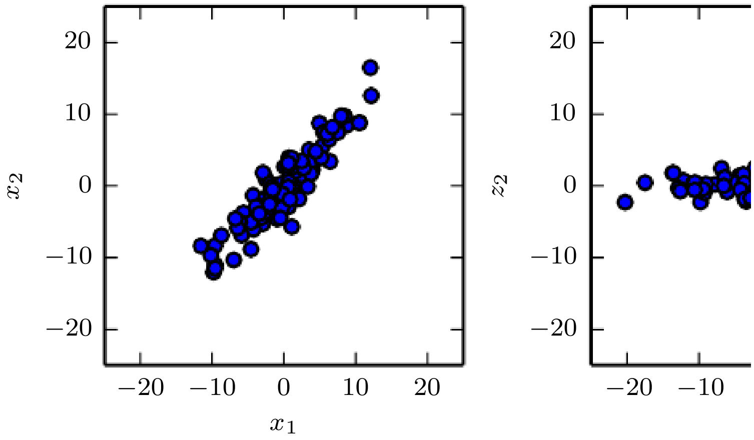


Figure 5.8: PCA learns a linear projection that aligns the direction
the axes of the new space. *(Left)*The original data consist of sampl
variance might occur along directions that are not axis aligned.
data $\boldsymbol{z} = \boldsymbol{x}^\top \boldsymbol{W}$ now varies most along the axis z_1. The direction
is now along z_2.

5.8.1 Principal Components Analysis

In section 2.12, we saw that the principal components anal
a means of compressing data. We can also view PCA as an
algorithm that learns a representation of data. This repr
two of the criteria for a simple representation described
representation that has lower dimensionality than the origin
a representation whose elements have no linear correlation
is a first step toward the criterion of learning representatio
statistically independent. To achieve full independence, a

a mean of zero, $\mathbb{E}[\boldsymbol{x}] = \mathbf{0}$. If this is not the case, the data
by subtracting the mean from all examples in a preprocess

The unbiased sample covariance matrix associated witl

$$\mathrm{Var}[\boldsymbol{x}] = \frac{1}{m-1}\boldsymbol{X}^\top\boldsymbol{X}.$$

PCA finds a representation (through linear transformati
$\mathrm{Var}[\boldsymbol{z}]$ is diagonal.

In section 2.12, we saw that the principal components
are given by the eigenvectors of $\boldsymbol{X}^\top\boldsymbol{X}$. From this view,

$$\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^\top.$$

In this section, we exploit an alternative derivation of the
The principal components may also be obtained via singul
(SVD). Specifically, they are the right singular vectors of $\boldsymbol{X}$
the right singular vectors in the decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}$
the original eigenvector equation with $\boldsymbol{W}$ as the eigenvect

$$\boldsymbol{X}^\top\boldsymbol{X} = \left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top\right)^\top \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top = \boldsymbol{W}\boldsymbol{\Sigma}^2$$

The SVD is helpful to show that PCA results in a diag
SVD of $\boldsymbol{X}$, we can express the variance of $\boldsymbol{X}$ as:

$$\begin{aligned}\mathrm{Var}[\boldsymbol{x}] &= \frac{1}{m-1}\boldsymbol{X}^\top\boldsymbol{X} \\ &= \frac{1}{m-1}(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top)^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{W}^\top\end{aligned}$$

$$= \frac{1}{m-1} \boldsymbol{W}^\top \boldsymbol{W} \boldsymbol{\Sigma}^2 \boldsymbol{W}^\top \boldsymbol{W}$$
$$= \frac{1}{m-1} \boldsymbol{\Sigma}^2,$$

where this time we use the fact that $\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{I}$, again fro SVD.

The above analysis shows that when we project the dat transformation $\boldsymbol{W}$, the resulting representation has a diag (as given by $\boldsymbol{\Sigma}^2$), which immediately implies that the indiv mutually uncorrelated.

This ability of PCA to transform data into a representat are mutually uncorrelated is a very important property c example of a representation that attempts to *disentangle t variation* underlying the data. In the case of PCA, this d form of finding a rotation of the input space (described l principal axes of variance with the basis of the new represen with $\boldsymbol{z}$.

While correlation is an important category of dependen the data, we are also interested in learning representations complicated forms of feature dependencies. For this, we wil can be done with a simple linear transformation.

5.8.2 k-means Clustering

Another example of a simple representation learning algorith The k-means clustering algorithm divides the training set i may be captured by a single integer.

The k-means algorithm works by initializing k different ce
to different values, then alternating between two different s
In one step, each training example is assigned to cluster i,
the nearest centroid $\boldsymbol{\mu}^{(i)}$. In the other step, each centroid
mean of all training examples $\boldsymbol{x}^{(j)}$ assigned to cluster i.

One difficulty pertaining to clustering is that the clusterir
ill posed, in the sense that there is no single criterion tha
clustering of the data corresponds to the real world. We c
of the clustering, such as the average Euclidean distance f
to the members of the cluster. This enables us to tell hc
reconstruct the training data from the cluster assignments
well the cluster assignments correspond to properties of th
there may be many different clusterings that all correspond
of the real world. We may hope to find a clustering that rel
obtain a different, equally valid clustering that is not rele
example, suppose that we run two clustering algorithms on
images of red trucks, images of red cars, images of gray truc
cars. If we ask each clustering algorithm to find two cluste
find a cluster of cars and a cluster of trucks, while anothe
red vehicles and a cluster of gray vehicles. Suppose we alsc
algorithm, which is allowed to determine the number of clu
the examples to four clusters, red cars, red trucks, gray cars
new clustering now at least captures information about bot
lost information about similarity. Red cars are in a diffe
cars, just as they are in a different cluster from gray truc
clustering algorithm does not tell us that red cars are mo

attributes instead of just testing whether one attribute ma

5.9 Stochastic Gradient Descent

Nearly all of deep learning is powered by one very important **gradient descent** (SGD). Stochastic gradient descent i gradient descent algorithm introduced in section 4.3.

A recurring problem in machine learning is that large tra for good generalization, but large training sets are also expensive.

The cost function used by a machine learning algorithm sum over training examples of some per-example loss func negative conditional log-likelihood of the training data can

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\text{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)},$$

where L is the per-example loss $L(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\log p(y \mid \boldsymbol{x};$

For these additive cost functions, gradient descent requ

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}).$$

The computational cost of this operation is $O(m)$. As the tr billions of examples, the time to take a single gradient step long.

The insight of SGD is that the gradient is an expectatio

using examples from the minibatch $\mathbb{B}$. The stochastic grad
then follows the estimated gradient downhill:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g},$$

where ϵ is the learning rate.

Gradient descent in general has often been regarded as
the past, the application of gradient descent to nonconvex
was regarded as foolhardy or unprincipled. Today, we k
learning models described in part II work very well when
descent. The optimization algorithm may not be guarant
local minimum in a reasonable amount of time, but it often
of the cost function quickly enough to be useful.

Stochastic gradient descent has many important uses
deep learning. It is the main way to train large linear
datasets. For a fixed model size, the cost per SGD update
training set size m. In practice, we often use a larger model
increases, but we are not forced to do so. The number of up
convergence usually increases with training set size. How
infinity, the model will eventually converge to its best po
SGD has sampled every example in the training set. Increa
extend the amount of training time needed to reach the m
error. From this point of view, one can argue that the asyn
a model with SGD is $O(1)$ as a function of m.

Prior to the advent of deep learning, the main way to
was to use the kernel trick in combination with a linear mode
algorithms require constructing an $m \times m$ matrix $G_{i,j} = k(\boldsymbol{x}$

5.10 Building a Machine Learning Alg

Nearly all deep learning algorithms can be described as a fairly simple recipe: combine a specification of a datas optimization procedure and a model.

For example, the linear regression algorithm combines $\boldsymbol{X}$ and $\boldsymbol{y}$, the cost function

$$J(\boldsymbol{w}, b) = -\mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}} \log p_{\mathrm{model}}(y \mid \boldsymbol{x}$$

the model specification $p_{\mathrm{model}}(y \mid \boldsymbol{x}) = \mathcal{N}(y; \boldsymbol{x}^{\top}\boldsymbol{w} + b, 1)$, optimization algorithm defined by solving for where the gra using the normal equations.

By realizing that we can replace any of these component from the others, we can obtain a wide range of algorithms.

The cost function typically includes at least one term t process to perform statistical estimation. The most comm negative log-likelihood, so that minimizing the cost func likelihood estimation.

The cost function may also include additional terms, terms. For example, we can add weight decay to the linear to obtain

$$J(\boldsymbol{w}, b) = \lambda||\boldsymbol{w}||_2^2 - \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}} \log p_{\mathrm{model}}(y$$

This still allows closed form optimization.

If we change the model to be nonlinear, then most cost be optimized in closed form. This requires us to choose

In some cases, the cost function may be a function th evaluate, for computational reasons. In these cases, we c minimize it using iterative numerical optimization, as long a approximating its gradients.

Most machine learning algorithms make use of this reci be immediately obvious. If a machine learning algorithm see hand designed, it can usually be understood as using a specia models, such as decision trees and k-means, require special-c their cost functions have flat regions that make them inappro by gradient-based optimizers. Recognizing that most machi can be described using this recipe helps to see the different taxonomy of methods for doing related tasks that work for than as a long list of algorithms that each have separate ju

5.11 Challenges Motivating Deep Lear

The simple machine learning algorithms described in this c wide variety of important problems. They have not succeed the central problems in AI, such as recognizing speech or r

The development of deep learning was motivated in traditional algorithms to generalize well on such AI tasks.

This section is about how the challenge of generalizing to exponentially more difficult when working with high-dime the mechanisms used to achieve generalization in traditi are insufficient to learn complicated functions in high-dim spaces also often impose high computational costs. Deep le

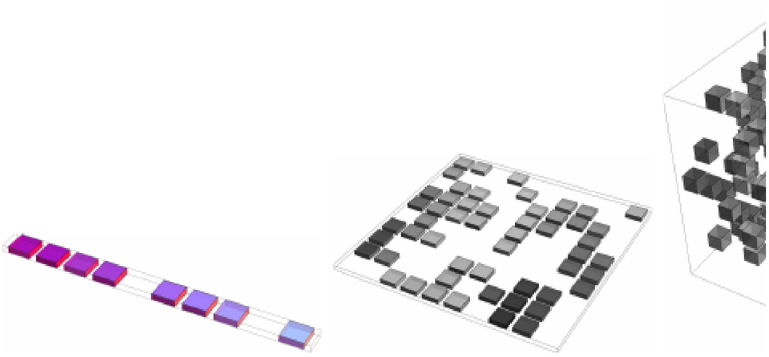


Figure 5.9: As the number of relevant dimensions of the datε right), the number of configurations of interest may grow exp one-dimensional example, we have one variable for which we on regions of interest. With enough examples falling within each of t corresponds to a cell in the illustration), learning algorithms can ϵ A straightforward way to generalize is to estimate the value of tl each region (and possibly interpolate between neighboring regi dimensions, it is more difficult to distinguish 10 different values c to keep track of up to 10 × 10 = 100 regions, and we need at leas cover all those regions. *(Right)*With three dimensions, this grow and at least that many examples. For d dimensions and v values each axis, we seem to need $O(v^d)$ regions and examples. This is of dimensionality. Figure graciously provided by Nicolas Chapa

One challenge posed by the curse of dimensionality is As illustrated in figure 5.9, a statistical challenge arises l possible configurations of $\boldsymbol{x}$ is much larger than the numbe To understand the issue, let us consider that the input sp grid, as in the figure. We can describe low-dimensional spac of grid cells that are mostly occupied by the data. When gen

algorithms simply assume that the output at a new point sh
the same as the output at the nearest training point.

5.11.2 Local Constancy and Smoothness Regu

To generalize well, machine learning algorithms need to be
about what kind of function they should learn. We have see
rated as explicit beliefs in the form of probability distributi
the model. More informally, we may also discuss prior beliefs
the *function* itself and influencing the parameters only indir
relationship between the parameters and the function. Add
discuss prior beliefs as being expressed implicitly by cho
are biased toward choosing some class of functions over
these biases may not be expressed (or even be possible to
probability distribution representing our degree of belief in

Among the most widely used of these implicit "priors
prior, or **local constancy prior**. This prior states that
should not change very much within a small region.

Many simpler algorithms rely exclusively on this prior
as a result, they fail to scale to the statistical challenges i
level tasks. Throughout this book, we describe how dee
additional (explicit and implicit) priors in order to redu
error on sophisticated tasks. Here, we explain why the smo
insufficient for these tasks.

There are many different ways to implicitly or explicitly
that the learned function should be smooth or locally const
methods are designed to encourage the learning process to le

the training set. For $k = 1$, the number of distinguishable r
than the number of training examples.

While the k-nearest neighbors algorithm copies the outpu
examples, most kernel machines interpolate between training
with nearby training examples. An important class of kernel
kernels, where $k(\boldsymbol{u}, \boldsymbol{v})$ is large when $\boldsymbol{u} = \boldsymbol{v}$ and decreases a
apart from each other. A local kernel can be thought of a
that performs template matching, by measuring how clo
resembles each training example $\boldsymbol{x}^{(i)}$. Much of the moder
learning is derived from studying the limitations of local t
how deep models are able to succeed in cases where local t
(Bengio *et al.*, 2006b).

Decision trees also suffer from the limitations of exclusi
learning, because they break the input space into as man
leaves and use a separate parameter (or sometimes many pa
of decision trees) in each region. If the target function r
least n leaves to be represented accurately, then at least n
required to fit the tree. A multiple of n is needed to achieve
confidence in the predicted output.

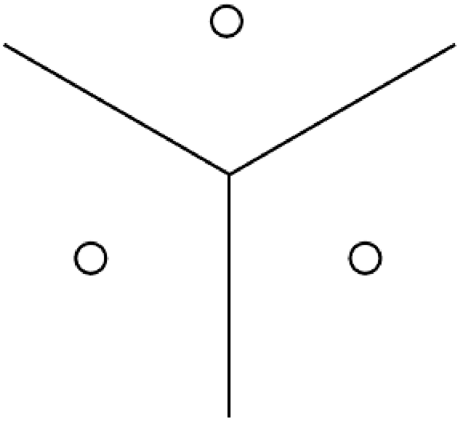
In general, to distinguish $O(k)$ regions in input space, al
$O(k)$ examples. Typically there are $O(k)$ parameters,
associated with each of the $O(k)$ regions. The nearest neigh
each training example can be used to define at most one r
figure 5.10.

Is there a way to represent a complex function that has
be distinguished than the number of training examples?
smoothness of the underlying function will not allow a le

least one example.

The smoothness assumption and the associated nonpa rithms work extremely well as long as there are enough exa algorithm to observe high points on most peaks and low] of the true underlying function to be learned. This is ge function to be learned is smooth enough and varies in fe In high dimensions, even a very smooth function can cha different way along each dimension. If the function addition in various regions, it can become extremely complicated to training examples. If the function is complicated (we want number of regions compared to the number of examples) generalize well?

The answer to both of these questions—whether it is a complicated function efficiently, and whether it is poss function to generalize well to new inputs—is yes. The key ins number of regions, such as $O(2^k)$, can be defined with $O(k)$



introduce some dependencies between the regions through a
about the underlying data-generating distribution. In this
generalize nonlocally (Bengio and Monperrus, 2005; Bengi
different deep learning algorithms provide implicit or explici
reasonable for a broad range of AI tasks in order to captur

Other approaches to machine learning often make stro
sumptions. For example, we could easily solve the checkerb
the assumption that the target function is periodic. Usually
strong, task-specific assumptions in neural networks so th
to a much wider variety of structures. AI tasks have struc
complex to be limited to simple, manually specified propert
so we want learning algorithms that embody more general
The core idea in deep learning is that we assume that the
the *composition of factors*, or features, potentially at mul
chy. Many other similarly generic assumptions can further
algorithms. These apparently mild assumptions allow an e
relationship between the number of examples and the num
be distinguished. We describe these exponential gains mo
6.4.1, 15.4 and 15.5. The exponential advantages conferr
distributed representations counter the exponential challen
of dimensionality.

5.11.3 Manifold Learning

An important concept underlying many ideas in machine
manifold.

A **manifold** is a connected region. Mathematically,

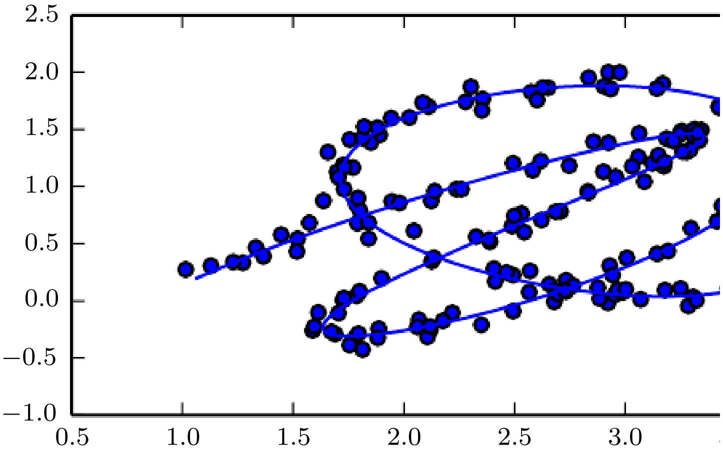
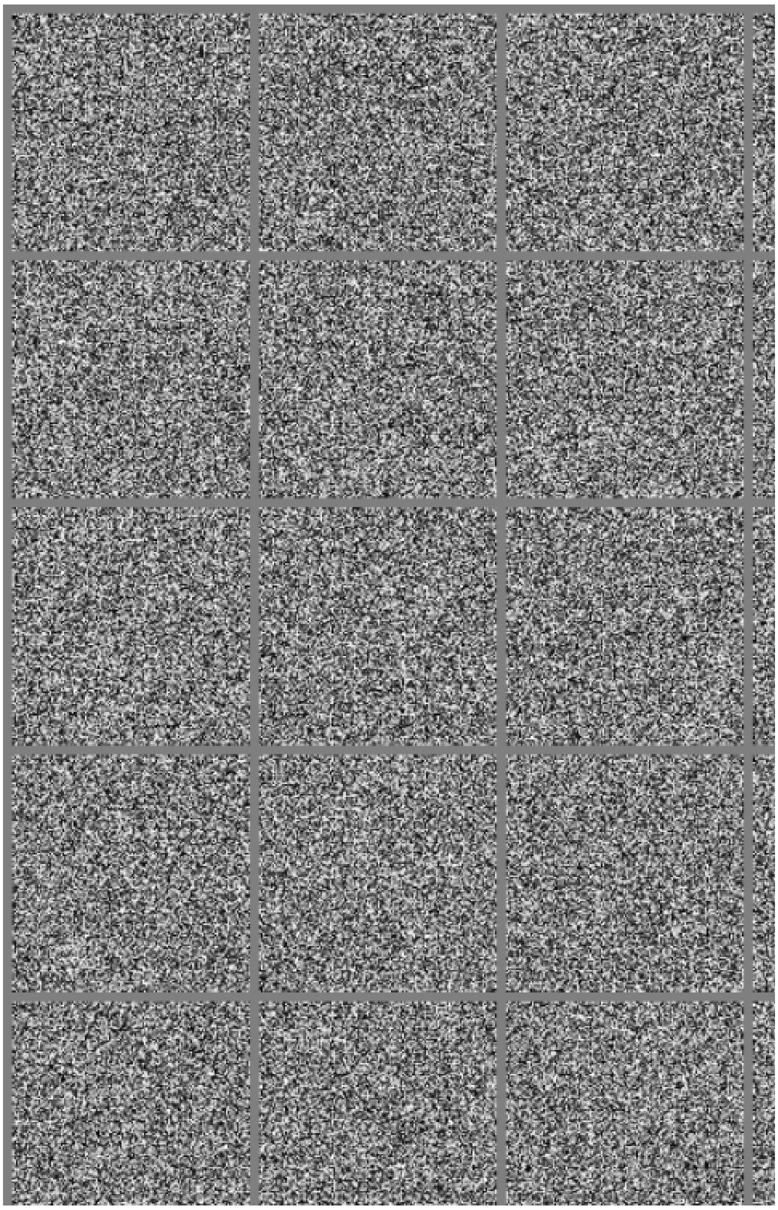


Figure 5.11: Data sampled from a distribution in a two-dimensio concentrated near a one-dimensional manifold, like a twisted strin the underlying manifold that the learner should infer.

degrees of freedom, or dimensions, embedded in a higher-di dimension corresponds to a local direction of variation. example of training data lying near a one-dimensional man dimensional space. In the context of machine learning, we a of the manifold to vary from one point to another. This manifold intersects itself. For example, a figure eight is a ma dimension in most places but two dimensions at the inters

Many machine learning problems seem hopeless if w learning algorithm to learn functions with interesting vari **Manifold learning** algorithms surmount this obstacle b of $\mathbb{R}^n$ consists of invalid inputs, and that interesting in



bility distribution over images, text strings, and sounds tl highly concentrated. Uniform noise essentially never resem from these domains. Figure 5.12 shows how, instead, unif look like the patterns of static that appear on analog televis is available. Similarly, if you generate a document by picki random, what is the probability that you will get a meanin text? Almost zero, again, because most of the long seque correspond to a natural language sequence: the distributic sequences occupies a very little volume in the total space c

Of course, concentrated probability distributions are not the data lies on a reasonably small number of manifolds. \ that the examples we encounter are connected to each otl with each example surrounded by other highly similar examp by applying transformations to traverse the manifold. Tl favor of the manifold hypothesis is that we can imagine su transformations, at least informally. In the case of images, of many possible transformations that allow us to trace ou space: we can gradually dim or brighten the lights, grac objects in the image, gradually alter the colors on the surf forth. Multiple manifolds are likely involved in most appl the manifold of human face images may not be connected face images.

These thought experiments convey some intuitive reason fold hypothesis. More rigorous experiments (Cayton, 2005; 2010; Schölkopf *et al.*, 1998; Roweis and Saul, 2000; Tenenba 2003; Belkin and Niyogi, 2003; Donoho and Grimes, 2003; 2004) clearly support the hypothesis for a large class of da

This concludes part I, which has provided the basic co
and machine learning that are employed throughout the r
book. You are now prepared to embark on your study of d