Overview of Machine Learning

Machine Learning, a Probabilistic Perspective

Chapter 1

Objective

Outcomes

- Formulate a well-posed learning problem for a real-world task by identifying the task, performance measure, and training experience
- Describe common learning paradigms in terms of the type of data available and when, the form of prediction, and the structure of the output prediction
- Identify examples of the ethical responsibilities of an ML expert

Questions

- What Does it Mean to Learn?
- What is the difference between memorization and generalization?

Evidence

- Ask students to formulate a learning problem (in class)
- Homework (focusing on revisions and assessing what they know in terms of basic probability and statistics, calculus, and linear algebra)

What Does it Mean to Learn?

The general supervised approach to machine learning: a learning algorithm reads in training data and computes a learned function f.

Outline

- 1.1 Machine learning: what and why?
  - 1.1.1 Types of machine learning
- 1.2 Supervised learning
  - 1.2.1 Classification
  - 1.2.2 Regression
- 1.3 Unsupervised learning
- 1.4 Some basic concepts in machine learning

1.1 Machine learning: what and why?

"We are drowning in information and starving for knowledge." — John Naisbitt.

The era of big data. This deluge of data calls for automated methods of data analysis, which is what **machine learning** provides.

Machine learning is defined as

- a **set of methods** that can
- **automatically detect patterns** in data, and then
- use the uncovered patterns to **predict future data**, or to
- perform other kinds of decision making under uncertainty

We are entering the era of big data . For example, there are about 1 trillion web pages; one hour of video is uploaded to YouTube every second, amounting to 10 years of content every day ;
the genomes of 1000s of people, each of which has a length of 3.8 ¡¿ 109 base pairs, have been sequenced by various labs; Walmart handles more than 1M transactions per hour and has databases containing more than 2.5 petabytes (2.5×10^{15}) of information (Cukier 2010); and so
on.

Examples of Machine Learning
https://www.redpixie.com/blog/examples-of-machine-learning

You can view ML's history by clicking on this link:
http://sge.wonderville.ca/machinelearning/history/history.html

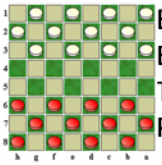
References:
http://www.contrib.andrew.cmu.edu/~mndarwis/ML.html
http://alex.smola.org/teaching/cmu2013-10-701/
https://classroom.udacity.com/courses/ud120/lessons/2410328539/concepts/24185

385370923
http://cs231n.github.io/

What is Machine Learning?

- In 1959 Arthur Samuel described it as:
  "**the field of study that gives computers the ability to learn without being explicitly programmed**."

- Tom Mitchell provides a more modern 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."



Example: playing checkers.
E = the experience of playing many games of checkers
T = the task of playing checkers.
P = the probability that the program will win the next game.

Humans learn from past experience
Computers follow instructions that have previously been programmed.
Can we get computers to learn from past experience too?

1.1 Machine learning: what and why? (cont.)

- The best way to solve such problems is to use the tools of **probability theory**.
- Probability theory can be applied to any problem involving uncertainty.
- In machine learning, uncertainty comes in many forms:
  - what is the best prediction about the future given some past data?
  - what is the best model to explain some data?
  - what measurement should I perform next? etc.

In ML we try to take as input **features** and we try to produce **labels**

1.1 Machine learning: what and why? (cont.)

- We will describe a wide variety of **probabilistic models**, suitable for a wide variety of data and tasks.
- We will also describe a wide variety of **algorithms for learning** and using such models.
- Goal: present a unified view of the field through the lens of **probabilistic modeling and inference**.

1.1.1 Types of machine learning

Machine learning is usually divided into two main types:

1. Predictive or **supervised** learning approach
   - Goal: **learn a mapping from inputs x to outputs y** , given a labeled set of input-output pairs: $D = \{x_i, y_i\}_{i=1}^{N}$
     - where D is called the **training set** , and
     - N is the number of training examples.
   - when y_i is categorical, the problem is known as **classification** or pattern recognition , and
   - when y_i is real-valued, the problem is known as **regression** .
2. Descriptive or **unsupervised** learning approach.
   - only given inputs pairs $D = \{x_i\}_{i=1}^{N}$ and the goal is to find "interesting patterns" in the data. This is sometimes called **knowledge discovery** .

Another variant of supervised learning, known as ordinal regression , occurs where label space Y has some natural ordering, such as grades A–F.

Terms

features , **attributes** or **covariates:**

each training input x_i is a D-dimensional vector of numbers, representing, say, the height and weight of a person.

however, x_i could be a complex structured object, such as an image, a sentence, an email message, a time series, a molecular shape, a graph, etc

output or **response variable:**

most methods assume that y_i is a categorical (or nominal) variable from some finite set, $y_i \in \{1, \ldots, C\}$ (such as male or female), or that y_i is a real-valued scalar (such as income level).

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

1.2 Supervised Learning

Regression problems. Predict results within a continuous output (i.e., map input variables to some continuous function.)

Classification problems. Predict results in a discrete output, (i.e., map input variables into discrete categories).

Example 1. Given data about the size of houses, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.

We could turn this example into a classification problem by instead making our output about whether the house "sells for more or less than the asking price." Here we are classifying the houses based on price into two discrete categories.

Example 2. (a) Regression - Given a picture of a person, predict their age

(b) Classification - Given a patient with a tumor, predict whether the tumor is malignant or benign.

1.2 Supervised Learning

1.2.1 Classification

Goal: **learn a mapping from inputs x to outputs y**,

where $y \in \{1, \ldots, C\}$, with C being the number of classes:

- if $C = 2$, this is called **binary classification** (in which case we often assume $y \in \{0, 1\}$);
- if $C > 2$, this is called **multiclass classification**.
- if the class labels are not mutually exclusive (e.g., somebody may be classified as tall and strong), we call it **multi-label classification**, but this is best viewed as predicting multiple related binary class labels (a so-called multiple output model).

Note: when we use the term "classification", we will mean multiclass classification with a single output, unless we state otherwise.

1.2 Supervised Learning

1.2.1 Classification (cont.)

One way to formalize the problem is as function approximation

assume y = f(x) for some unknown function f

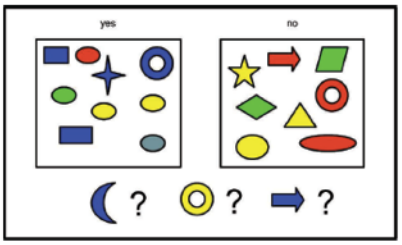
learning goal: estimate the function f given a labeled training set

then to make predictions using $\hat{y} = \hat{f}(x)$

(hat symbol denotes an estimate)

- Main goal is to make predictions on **novel** inputs, meaning ones that we have not seen before (this is called **generalization)**

1.2.1.1 Example



(a)

D features (attributes)

N cases

Color	Shape	Size (cm)	Label
Blue	Square	10	1
Red	Ellipse	2.4	1
Red	Ellipse	20.7	0

(b)

Figure 1.1 Left: Some labeled training examples of colored shapes, along with 3 unlabeled test cases. Right: Representing the training data as an $N \times D$ design matrix. Row i represents the feature vector $\mathbf{x}_i$. The last column is the label, $y_i \in \{0, 1\}$. Based on a figure by Leslie Kaelbling.

As a simple toy example of classification, consider the problem illustrated in Figure 1.1(a).
We have two classes of object which correspond to labels 0 and 1. The inputs are colored shapes.
These have been described by a set of D features or attributes, which are stored in an N x D design matrix X , shown in Figure 1.1(b). The input features x can be discrete, continuous or a combination of the two. In addition to the inputs, we have a vector of training labels y .

In Figure 1.1, the test cases are a blue crescent, a yellow circle and a blue arrow. None of these have been seen before. Thus we are required to generalize beyond the training set. A reasonable guess is that blue crescent should be y = 1 , since all blue shapes are labeled 1 in the
training set. The yellow circle is harder to classify, since some yellow things are labeled y = 1 and some are labeled y = 0 , and some circles are labeled y = 1 and some y = 0 . Consequently it is not clear what the right label should be in the case of the yellow circle. Similarly, the correct
label for the blue arrow is unclear.

1.2.1.2 The need for probabilistic predictions

To handle ambiguous cases, such as the yellow circle above, it is desirable to return a probability.

$p(y|x, \mathcal{D})$ denotes the probability distribution over possible labels, given the input vector x and training set $\mathcal{D}$

- $p(y|x, \mathcal{D})$ represents a vector of length C (where C is the number of classes)
- for two classes, it is sufficient to return the single number $p(y = 1|x, \mathcal{D})$, since $p(y = 1|x, \mathcal{D}) + p(y = 0|x, \mathcal{D}) = 1$
- the probability is conditional on the test input x, as well as the training set $\mathcal{D}$, by putting these terms on the right hand side of the conditioning bar |

To handle ambiguous cases, such as the yellow circle above, it is desirable to return a probability.

1.2.1.2 The need for probabilistic predictions (cont.)

- Given a probabilistic output, we can always compute our "best guess" as to the "true label" using: $\hat{y} = \hat{f}(\mathbf{x}) = \underset{c=1}{\overset{C}{\operatorname{argmax}}}\, p(y = c|\mathbf{x}, \mathcal{D})$
- This corresponds to the most probable class label, and is called the mode of the distribution
- p(y|x,D) ; it is also known as a MAP estimate (MAP stands for maximum a posteriori).

1.2.1.3 Real-world applications

https://www.what-dog.net/

1.2.1.3 Real-world applications

Classifying flowers

Figure 1.3 Three types of iris flowers: setosa, versicolor and virginica. Source: `http://www.statlab.uni-heidelberg.de/data/iris/` . Used with kind permission of Dennis Kramb and SIGNA.

The goal is to learn to distinguish three different kinds of iris flower.

https://github.com/probml/pyprobml/blob/master/examples/fisheririsDemo.py

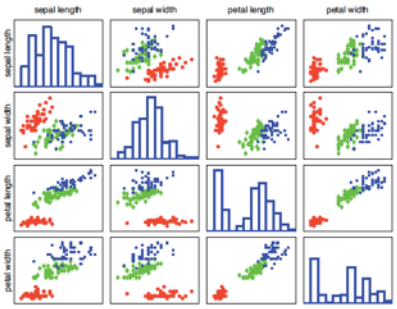


Figure 1.4 Visualization of the Iris data as a pairwise scatter plot. The diagonal plots the marginal histograms of the 4 features. The off diagonals contain scatterplots of all possible pairs of features. Red circle = setosa, green diamond = versicolor, blue star = virginica. Figure generated by `fisheririsDemo`.

1.2.2 Regression

- Regression is just like classification except the response variable is continuous.
- we have a single real-valued input $x_i \in R$, and a single real-valued response yi $\in R$. We consider fitting two models to the data: a straight line and a quadratic function.

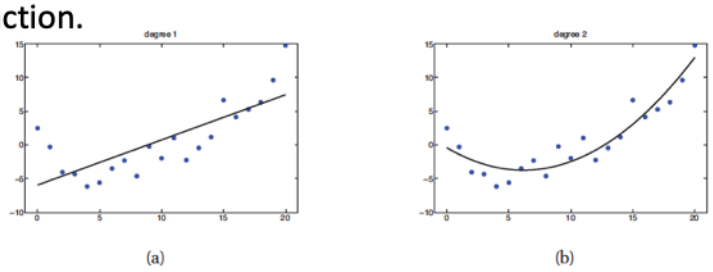


Figure 1.7 (a) Linear regression on some 1d data. (b) Same data with polynomial regression (degree 2). Figure generated by `linregPolyVsDegree`.

(We explain how to fit such models later.) Various extensions of this basic problem can arise, such as having high-dimensional inputs, outliers, non-smooth responses, etc. We will discuss ways to handle such problems later in the book.

Examples of real-world regression problems.

- Predict the age of a viewer watching a given video on YouTube.
- Predict the location in 3d space of a robot arm end effector, given control signals (torques) sent to its various motors.
- Predict the temperature at any location inside a building using weather data, time, door sensors, etc.

1.3 Unsupervised learning

- The goal is to discover "interesting structure" in the data, where we are just given output data, without any inputs; this is sometimes called **knowledge discovery** .
- We will formalize our task as one of density estimation, that is, we want to build models of the form $p(x_i|\theta)$.
- There are two differences from the supervised case.
  - First, we have written $p(x_i|\theta)$ instead of $p(y_i|x_i, \theta)$; i.e., supervised learning is conditional density estimation, whereas unsupervised learning is unconditional density estimation.
  - Second, x_i is a vector of features, so we need to create multivariate probability models.

By contrast, in supervised learning, yi is usually just a single variable that we are trying to predict. This means that for most supervised learning problems, we can use univariate probability models (with input-dependent parameters), which significantly simplifies the problem.

1.3 Unsupervised learning

1.3.1 Discovering clusters

- As a canonical example of unsupervised learning, consider the problem of clustering data into groups. Consider the following data, representing the height and weight of a group of 210 people.

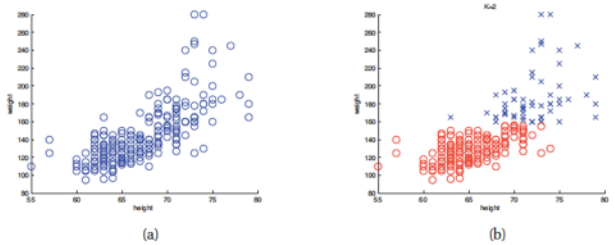


Figure 1.8 (a) The height and weight of some people. (b) A possible clustering using $K = 2$ clusters. Figure generated by `kmeansHeightWeight`.

https://github.com/probml/pyprobml/blob/master/examples/kmeansHeightWeight.py

1.3 Unsupervised learning

1.3.1 Discovering clusters

Let K denote the number of clusters.

1st goal: estimate the distribution over the number of clusters, p(K|D) ; this tells us if there are subpopulations within the data.

For simplicity, we often approximate the distribution p(K|D) by its mode,

$K^* = \arg\max_K p(K|D)$

2nd goal: estimate which cluster each point belongs to.

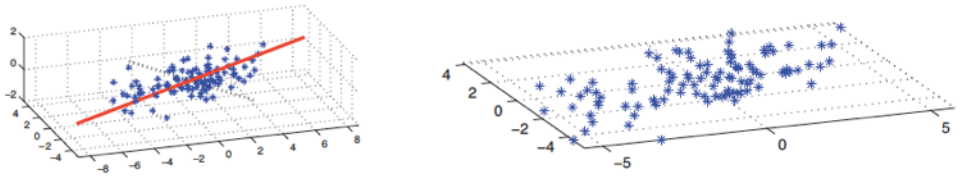
Let $z_i \in \{1, \ldots, K\}$ represent the cluster to which data point i is assigned. (z_i is an example of a hidden or latent variable, since it is never observed in the training set.)

We can infer which cluster each data point belongs to by computing

$$z_i^* = argmax_k p(z_i = k|x_i, D)$$

1.3.2 Discovering latent factors

- When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the "essence" of the data. This is called **dimensionality reduction**.



(a) (b)

Figure 1.9 (a) A set of points that live on a 2d linear subspace embedded in 3d. The solid red line is the first principal component direction. The dotted black line is the second PC direction. (b) 2D representation of the data. Figure generated by `pcaDemo3d`.

A simple example is shown in Figure 1.9, where we project some 3d data down to a 2d plane. The 2d approximation is quite good, since most points lie close to this subspace. Reducing to 1d would involve projecting points onto the red line in Figure 1.9(a); this would be a rather poor approximation. (We will make this notion precise in Chapter 12.)

Outline

- 1.4 Some basic concepts in machine learning
  - 1.4.1 Parametric vs non-parametric models
  - 1.4.2 A simple non-parametric classifier: K -nearest neighbors
  - 1.4.3 The curse of dimensionality
  - 1.4.4 Parametric models for classification and regression
  - 1.4.5 Linear regression
  - 1.4.6 Logistic regression
  - 1.4.7 Overfitting
  - 1.4.8 Model selection

1.4 Some basic concepts in machine learning

1.4.1 Parametric vs non-parametric models

- Parametric model: fixed number of parameters
- Nonparametric: the number of parameters grow with the amount of training data.
- Parametric models have the advantage of often being faster to use, but the disadvantage of making stronger assumptions about the nature of the data distributions.
- Nonparametric models are more flexible, but often computationally intractable for large datasets.

Other references for parametric vs non-parametric
http://mlss.tuebingen.mpg.de/2015/slides/ghahramani/gp-neural-nets15.pdf

1.4.2 A simple non-parametric classifier: K - nearest neighbors

- A simple example of a non-parametric classifier is the K nearest neighbor (KNN) classifier.
- This simply "looks at" the K points in the training set that are nearest to the test input x , counts how many members of each class are in this set, and returns that empirical fraction as the estimate

$$p(y = c|\mathbf{x}, \mathcal{D}, K) = \frac{1}{K} \sum_{i \in N_K(\mathbf{x}, \mathcal{D})} \mathbb{I}(y_i = c) \tag{1.2}$$

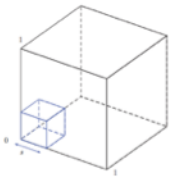
where $N_K(\mathbf{x}, \mathcal{D})$ are the (indices of the) K nearest points to $\mathbf{x}$ in $\mathcal{D}$ and $\mathbb{I}(e)$ is the **indicator function** defined as follows:

$$\mathbb{I}(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases} \tag{1.3}$$

1.4.3 The curse of dimensionality

- The KNN classifier is simple and can work quite well, provided it is **given a good distance metric**
- and has **enough labeled training data**. In fact, it can be shown that the KNN classifier can come within a factor of 2 of the best possible performance if $N \rightarrow \infty$ (Cover and Hart 1967).
- However, the main problem with KNN classifiers is that they do not work well with high dimensional inputs. The poor performance in high dimensional settings is due to the curse of dimensionality .

1.4.3 The curse of dimensionality (cont.)



Consider applying a KNN classifier to data where the inputs are uniformly distributed in the D -dimensional unit cube.

Suppose we estimate the density of class labels around a test point x by "growing" a hyper-cube around x until it contains a desired fraction f of the data points.

The expected edge length of this cube will be $e_D(f) = f1/D$. If D = 10 , and we want to base our estimate on 10% of the data, we have $e_{10}(0.1) = 0.8$, so we need to extend the cube 80% along each dimension around x .

Even if we only use 1% of the data, we find $e_{10}(0.01) = 0.63$

see Figure 1.16. Since the entire range of the data is only 1 along each dimension, we see that the method is no longer very local, despite the name "nearest neighbor". The trouble with looking at neighbors that are
so far away is that they may not be good predictors about the behavior of the input-output
function at a given point.

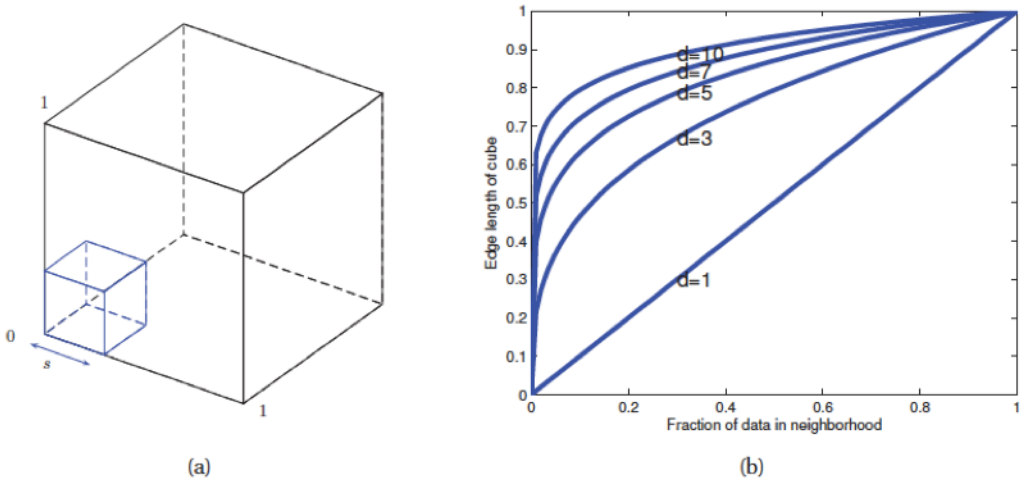


Figure 1.16 Illustration of the curse of dimensionality. (a) We embed a small cube of side s inside a larger unit cube. (b) We plot the edge length of a cube needed to cover a given volume of the unit cube as a function of the number of dimensions. Based on Figure 2.6 from (Hastie et al. 2009). Figure generated by `curseDimensionality`.

see Figure 1.16. Since the entire range of the data is only 1 along each dimension, we see that the method is no longer very local, despite the name "nearest neighbor". The trouble with looking at neighbors that are
so far away is that they may not be good predictors about the behavior of the input-output
function at a given point.

1.4.5 Linear regression

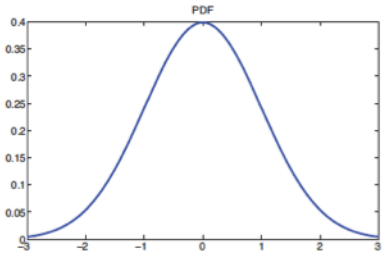
- One of the most widely used models for regression is known as **linear regression**.
- This asserts that the response is a linear function of the inputs.
- This can be written as follows:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon$$

- where $\mathbf{w}^T\mathbf{x}$ represents the inner or **scalar product** between the input vector x and the model's **weight vector** w, and ε is the **residual error** between our linear predictions and the true response.

7. In statistics, it is more common to denote the regression weights by β.

1.4.5 Linear regression (cont.)



We often assume that ϵ has a **Gaussian**[8] or **normal** distribution. We denote this by $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$, where μ is the mean and σ^2 is the variance (see Chapter 2 for details). When we plot this distribution, we get the well-known **bell curve** shown in Figure 1.17(a).

To make the connection between linear regression and Gaussians more explicit, we can rewrite the model in the following form:

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mu(\mathbf{x}), \sigma^2(\mathbf{x})) \tag{1.5}$$

This makes it clear that the model is a conditional probability density. In the simplest case, we assume μ is a linear function of $\mathbf{x}$, so $\mu = \mathbf{w}^T\mathbf{x}$, and that the noise is fixed, $\sigma^2(x) = \sigma^2$. In this case, $\boldsymbol{\theta} = (\mathbf{w}, \sigma^2)$ are the parameters of the model.

probability density function (**PDF**), or **density** of a continuous random variable, is a function, whose value at any given sample (or point) in the sample space (the set of possible values taken by the random variable) can be interpreted as providing a *relative likelihood* that the value of the random variable would equal that sample.

1.4.5 Linear regression (cont.)

For example, suppose the input is 1 dimensional. We can represent the expected response as follows:

$$\mu(\mathbf{x}) = w_0 + w_1 x = \mathbf{w}^T \mathbf{x}$$

where,

w_0 is the intercept or bias term,

w1 is the slope, and where

we have defined the vector $\mathbf{x} = (1, \tilde{x})$ (Prepending a constant 1 term to an input vector allows us to combine the intercept term with the other terms)

If w1 is positive, it means we expect the output to increase as the input increases.

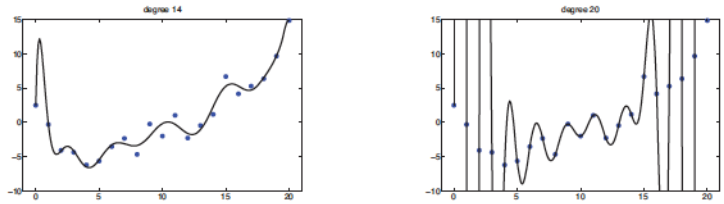
1.4.5 Linear regression (cont.)

Linear regression can be made to model non-linear relationships by replacing $\mathbf{x}$ with some non-linear function of the inputs, $\phi(\mathbf{x})$. That is, we use

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T \phi(\mathbf{x}), \sigma^2) \tag{1.7}$$

This is known as **basis function expansion**. For example, Figure 1.18 illustrates the case where $\phi(\mathbf{x}) = [1, x, x^2, \ldots, x^d]$, for $d = 14$ and $d = 20$; this is known as **polynomial regression**. We will consider other kinds of basis functions later in the book. In fact, many popular machine learning methods — such as support vector machines, neural networks, classification and regression trees, etc. — can be seen as just different ways of estimating basis functions from data, as we discuss in Chapters 14 and 16.

1.4.5 Linear regression (cont.)



(a) (b)

Figure 1.18 Polynomial of degrees 14 and 20 fit by least squares to 21 data points. Figure generated by `linregPolyVsDegree`.

1.4.6 Logistic regression

- We can generalize linear regression to the (binary) classification setting by making two changes:
  1. replace the Gaussian distribution for y with a Bernoulli distribution, which is more appropriate for the case when the response is binary, $y \in \{0, 1\}$. That is, we use $p(y|x,w) = \text{Ber}(y|\mu(x))$, where $\mu(x) = E[y|x] = p(y = 1|x)$.
  2. compute a linear combination of the inputs, as before, but then we pass this through a function that ensures $0 \leq \mu(x) \leq 1$ by defining $\mu(x) = \text{sigm}(w^T x)$, where $\text{sigm}(\eta)$ refers to the **sigmoid** function (also known as the **logistic** or **logit** function).
- This is defined as: $\text{sigm}(\eta) \triangleq \frac{1}{1+\exp(-\eta)} = \frac{e^\eta}{e^\eta + 1}$

1.4.6 Logistic regression (cont.)

- The term "sigmoid" means S-shaped. It is also known as a squashing function , since it maps the whole real line to [0, 1] , which is necessary for the output to be interpreted as a probability.
- Putting these two steps together we get

$$p(y|x,w) = \text{Ber}(y|\text{sigm}(w^T x))$$

- This is called **logistic regression** due to its similarity to linear regression
  - (although it is a form of classification, not regression!).

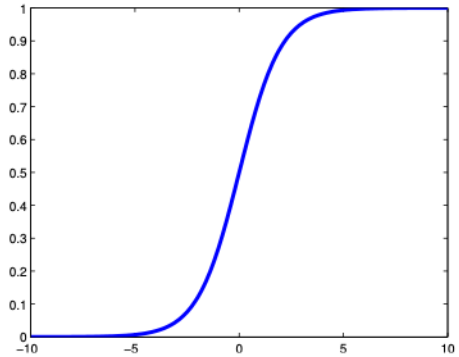


Figure 1.19 (a) The sigmoid or logistic function. We have sigm(−∞) = 0, sigm(0) = 0.5, and sigm(∞) = 1. Figure generated by sigmoidPlot.

1.4.6 Logistic regression (cont.)

- A simple example of logistic regression is shown in Figure 1.19(b), where we plot

$$p(y_i = 1 | x_i, w) = \text{sigm}(w_0 + w_1 x_i)$$

- where x_i is the SAT score of student i and y_i is whether they passed or failed a class.

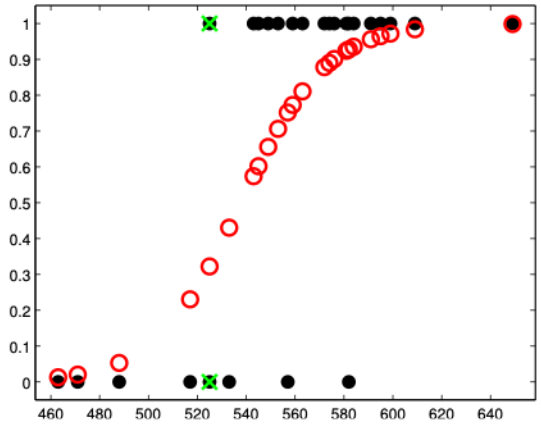


Figure 1.19 (b) Logistic regression for SAT scores. Solid black dots are the data. The open red circles are the predicted probabilities. The green crosses denote two students with the same SAT score of 525 (and hence same input representation x) but with different training labels (one student passed, y = 1, the other failed, y = 0). Hence this data is not perfectly separable using just the SAT feature. Figure generated by logregSATdemo.

1.4.6 Logistic regression (cont.)

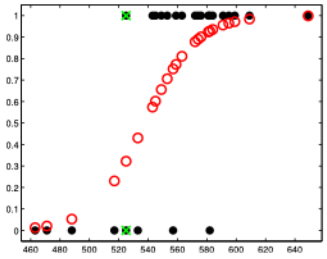
- The solid black dots show the training data, and the red circles plot $p(y = 1|x_i, \hat{w})$, where $\hat{w}$ are the parameters estimated from the training data (we discuss how to compute these estimates in Section 8.3.4).
- If we threshold the output probability at 0.5, we can induce a decision rule of the form

$$\hat{y}(x) \Leftrightarrow p(y = 1|x) > 0.$$

1.4.6 Logistic regression (cont.)

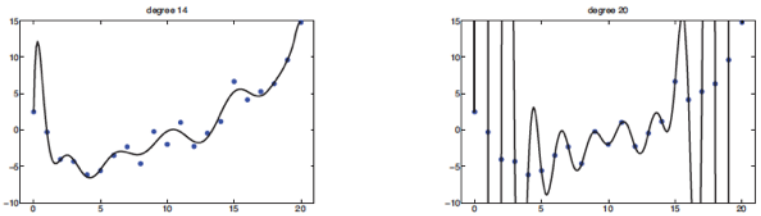
By looking at Figure 1.19(b), we see that $\text{sigm}(w_0 + w_1 x) = 0.5$ for $x \approx 545 = x^*$. We can imagine drawing a vertical line at $x = x^*$; this is known as a decision boundary. Everything to the left of this line is classified as a 0, and everything to the right of the line is classified as a 1.

We notice that this decision rule has a non-zero error rate even on the training set. This is because the data is not **linearly separable**, i.e., there is no straight line we can draw to separate the 0s from the 1s. We can create models with non-linear decision boundaries using basis function expansion, just as we did with non-linear regression. We will see many examples of this later in the book.



1.4.7 Overfitting

- When we fit highly flexible models, we need to be careful that we do not overfit the data, that is, we should avoid trying to model every minor variation in the input, since this is more likely to be noise than true signal.



(a) (b)

Figure 1.18 Polynomial of degrees 14 and 20 fit by least squares to 21 data points. Figure generated by `linregPolyVsDegree`.

This is illustrated in Figure 1.18(b), where we see that using a high degree polynomial results in a curve that is very "wiggly". It is unlikely that the true function has such extreme oscillations. Thus using such a model might result in accurate predictions of future outputs.

1.4.8 Model selection

- When we have a variety of models of different complexity (e.g., linear or logistic regression models with different degree polynomials, or KNN classifiers with different values of K), how should we pick the right one?
- A natural approach is to compute the **misclassification rate** on the training set for each method.
- This is defined as follows:

where f(x) is our classifier.

$$\mathrm{err}(f, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(f(\mathbf{x}_i) \neq y_i)$$

1.4.8 Model selection (cont.)

Figure 1.21(a) shows:
error rate vs K for a KNN classifier (dotted blue)
test error vs K in solid red (upper curve)

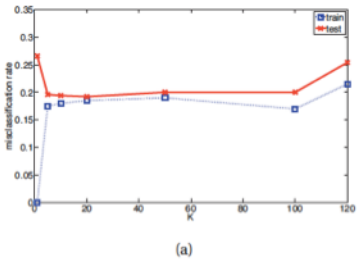


Figure 1.21 (a) Misclassification rate vs K in a K-nearest

However, what we care about is generalization error, which is the expected value of the misclassification rate when averaged over future data (see Section 6.3 for details).

This can be approximated by computing the misclassification rate on a large independent test set , not used during model training.

Now we see a U-shaped curve : for complex models (small K), the method overfits, and for
simple models (big K), the method underfits . Therefore, an obvious way to pick K is to pick
the value with the minimum error on the test set (in this example, any value between 10 and
100 should be fine).

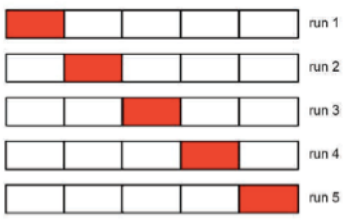
1.4.8 Model selection (cont.)

- Unfortunately, when training the model, we don't have access to the test set, so we cannot use the test set to pick the model of the right complexity.
- However, we can create a test set by partitioning the training set into two: the part used for training the model, and a second part, called the validation set , used for selecting the model complexity.
- We then fit all the models on the training set, and evaluate their performance on the validation set, and pick the best.
- Often we use about 80% of the data for the training set, and 20% for the validation set.

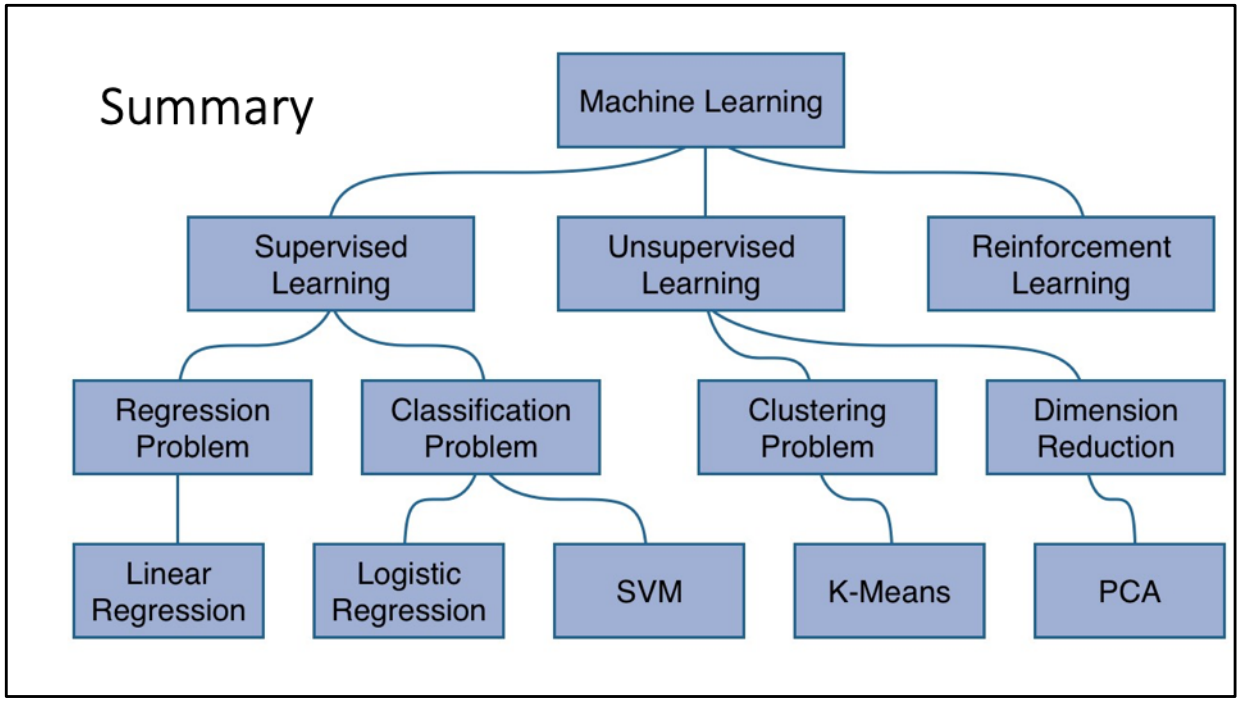
1.4.8 Model selection (cont.)

cross validation (CV).

- split the training data into K folds ; then, for each fold $k \in \{1, \ldots, K\}$, train on all the folds but the k 'th, and test on the k 'th, in a round-robin fashion, as sketched in Figure 1.21(b).
- We then compute the error averaged over all the folds, and use this as a proxy for the test error. (Note that each point gets predicted only once, although it will be used for training K−1 times.)
- It is common to use K = 5 ; this is called 5-fold CV estimate of the test error vs K , and to compare it to the empirical test error in Figure 1.21(a)



If we set K = N , then we get a method
called leave-one out cross validation , or LOOCV , since in fold i , we train on all the data cases
except for i , and then test on i . Exercise 1.3 asks you to compute the 5-fold CV estimate of the
test error vs K , and to compare it to the empirical test error in Figure 1.21(a).


Summary
Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Regression Problem
Classification Problem
Clustering Problem
Dimension Reduction
Linear Regression
Logistic Regression
SVM
K-Means
PCA

Summary (cont.)

Other take home terms:

- Linearly separable dataset
- Overfitting
- Cross validation