5. Feature spaces

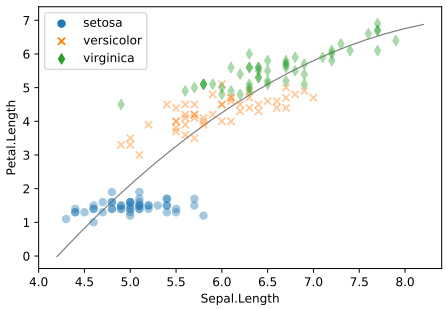
Goals. Refresh your memory of IA Maths for NST, where you were taught about linear spaces, bases, inner products, and projections. Understand the link between linear regression and inner products. Get experience of feature engineering.

In data science, a *feature* is any measurable property of the objects being studied. A *linear model* is a model with unknown parameters in which the parameters are weighted by features and combined linearly. Here's a very simple example.

The Iris dataset was collected by the botanist Edgar Anderson and popularized[32] by Ronald Fisher in 1936. Fisher has been described as a "genius who almost single-handedly created the foundations for modern statistical science". The dataset consists of 50 samples from each of three species of iris, each with four measurements.

Petal length	Petal width	Sepal length	Sepal width	species
1.0	0.2	4.6	3.6	setosa
5.0	1.9	6.3	2.5	virginica
5.8	1.6	7.2	3.0	virginica
1.7	0.5	5.1	3.3	setosa
4.2	1.2	5.7	3.0	versicolor
⋮				

Suppose we're interested in how petal length depends on sepal length. Here is a plot:



The plot also shows a smooth curve for the fitted model

$$\mathsf{Petal.Length} \approx \alpha + \beta\,\mathsf{Sepal.Length} + \gamma\,(\mathsf{Sepal.Length})^2. \tag{23}$$

By *fitted model* we mean that the parameters α, β and γ have been chosen so as to make the approximation as good as possible. What does this mean? In Section 5.4 we will see how to formulate models like (23) probabilistically. This will let us be precise about what the best fit is (namely, the maximum likelihood fit), and it will also let us do inference (e.g. report confidence intervals for the parameters.) But for now here is the code:

```
# columns from the dataset
x,y = data['Sepal.Length'], data['Petal.Length']
# Fit the model: takes a feature matrix, and a col. vector of outcomes
model = sklearn.linear_model.LinearRegression()
```

[32] It's tempting for computer scientists and mathematicians to think that data science is about algorithms and calculating with distributions and so on, but shared datasets are arguably more important. C.P. Scott, the former editor of *The Guardian*, said "Comment is free, but facts are sacred".

Modern advances in neural networks and deep learning were propelled by two shared datasets: the MNIST database of handwritten digits, and the ImageNet database of labelled photos. The story of ImageNet and of Fei-Fei Li, the researcher who collected it, is told in *The data that transformed AI research—and possibly the world*, `https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/`.

In addition to shared datasets, it's also useful to have a shared challenge, what David Donoho calls a *common task framework*. See David Donoho. *50 years of Data Science*. Presentation at the Tukey centennial workshop. 2015. URL: `http://courses.csail.mit.edu/18.337/2015/docs/50YearsDataScience.pdf`

```
model.fit(np.column_stack([x, x**2]), y[:, np.newaxis])
# plot a curve to depict the fitted values from the model
newx = np.linspace(4.2, 8.2, 200)
plt.plot(newx, model.predict(np.column_stack([newx, newx**2])))
```

In this model, we would say there are two features, Sepal.Length and $(\text{Sepal.Length})^2$. The dataset has other attributes, and they can be transformed to create an infinite variety of features, but we only use the word *feature* for data attributes that are being used in a model. We call Petal.Length the *outcome* or *label* of this model, not a feature.

Why two features, and not one or three? From the perspective of the person preparing the dataset, there is only one feature, Sepal.Length. From the perspective of the person invoking model.fit(), there are two data features that have to be passed in, and it's irrelevant that they came from the same column in the dataset. From the perspective of the person who implements model.fit(), there is a third feature, the constant feature that is being used as a weight for the α parameter. (The model.fit() command automatically adds this feature for you, unless you tell it not to by model.fit(..., fit_intercept=False).) Don't get uptight about defining the word 'feature', just write out your models explicitly, and there will be no confusion.

* * *

The model is linear because it combines the unknown parameters α, β and γ in a linear formula. There's no reason to think this is in any way a 'true' model, and we could equally well have proposed a non-linear model e.g.

$$\text{Petal.Length} \approx \alpha - \beta e^{-\gamma \text{Sepal.Length}}.$$

Linear models are especially convenient to work with. They lend themselves to efficient algorithms; linear models strained the computing capabilities of a desktop PC in the late 1980s, and non-linear models in the form of neural networks strain the computing capabilities of clusters of GPUs in server farms today.

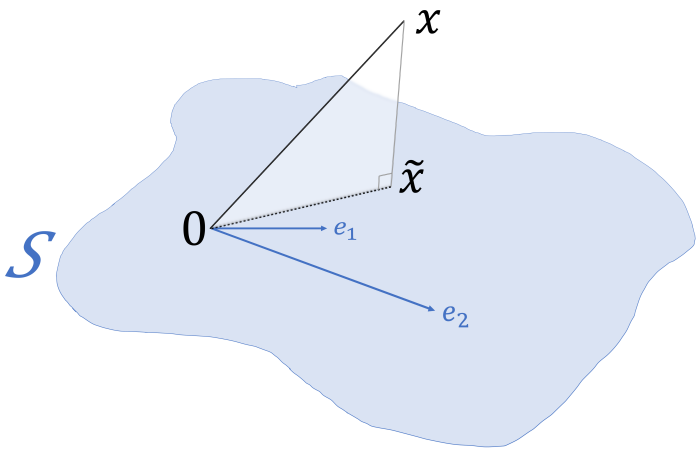
Linear models are also an easy way to explore aspects of the dataset, and they are the building block for many other models, some of which you will study in Part II *Machine Learning and Bayesian Inference*: support vector machines, perceptrons, and deep neural networks.

The intuition behind linear modelling comes from the mathematics of linear spaces. We'll revise this maths first, before returning to linear modelling.

5.1. Linear mathematics

This picture illustrates all the concepts from vector spaces and linear mathematics that we'll need for this data science course:

- Linearly independent basis vectors e_1 and e_2
- The linear subspace spanned by those vectors, $S = \{\lambda_1 e_1 + \lambda_2 e_2 \,:\, -\infty < \lambda_1, \lambda_2 < \infty\}$
- Another vector x can be projected onto the subspace, by finding the point $\tilde{x} = \hat{\lambda}_1 e_1 + \hat{\lambda}_2 e_2$ in S that is closest to x
- The residual $x - \tilde{x}$ is orthogonal to the basis vectors



We'll now define some these concepts abstractly and mathematically (leaving projections to Section 5.4). It's good to get intuition from three dimensional Euclidean space—but it's also useful to have abstract definitions so that the concepts can be applied to more general settings, as you will see in Part II *Digital Signal Processing* and *Computer Vision* (Fourier transforms and wavelets) and *Quantum Computing*.

5.1.1. ABSTRACT DEFINITIONS

- Let V be a set whose elements are called *vectors*, denoted by Roman letters u, v, w, etc.
- Let F be a field whose elements are called *scalars*, denoted by Greek letters λ, μ, etc. For our purposes, take F to be either the real numbers or the complex numbers.
- Let there be a binary operation $V \times V \to V$, called *addition*, written $v + w$.
- Let there be a binary operation $F \times V \to V$, called *scalar multiplication*, written λv.
- Let there be a binary operation $V \times V \to F$, called *inner product*, written $v \cdot w$.

Vector space. V is called a *vector space* over F if the following properties hold:

1. Associativity: $(u + v) + w = u + (v + w)$ for all vectors u, v, w.
2. Commutativity: $u + v = v + u$ for all vectors u, v
3. Zero vector: there is a vector 0 such that $v + 0 = v$ for all vectors v
4. Inverse: for every vector v there is a vector denoted $-v$ such that $v + (-v) = 0$
5. $\lambda(v + w) = \lambda v + \lambda w$ for every scalar λ and vectors v, w
6. $(\lambda + \mu)v = \lambda v + \mu v$ and $(\lambda\mu)v = \lambda(\mu v)$ for all scalars λ, μ and vector v
7. $1v = v$ for every vector v, where 1 is the unit scalar (i.e. $1\lambda = \lambda$ for every scalar λ).

In introductory geometry it's common to use bold symbols for vectors, e.g. $\mathbf{v} + \mathbf{0} = \mathbf{v}$ and $1\mathbf{v} = \mathbf{v}$. This notation makes it clear that $\mathbf{0}$ is a vector and 1 is a scalar. The bold notation is less common in more advanced applications, so you have to rely on type inference to spot that 0 is a vector and 1 is a scalar.

Linear combinations and bases. Let $v_1, \ldots, v_n$ be vectors in a vector space and $\lambda_1, \ldots, \lambda_n$ be scalars. Then the vector $\lambda_1 v_1 + \cdots + \lambda_n v_n$ is called a *linear combination* of $v_1, \ldots, v_n$. The set of all linear combinations

$$S = \{\lambda_1 v_1 + \cdots + \lambda_n v_n \,:\, \lambda_i \in F \text{ for all } i\}$$

is called the *span* of $\{v_1, \ldots, v_n\}$, and the vectors v_i are said to *span* S. Clearly $S \subseteq V$, and it is not hard to check that S is also a vector space. It is called a *subspace* of V.

Vectors $v_1, \ldots, v_n$ in a vector space are said to be *linearly independent* if

$$\lambda_1 v_1 + \cdots + \lambda_n v_n = 0 \quad \Longrightarrow \quad \lambda_1 = \cdots = \lambda_n = 0.$$

If this is not the case, then they are said to be *linearly dependent*.

If there is a finite set of vectors $e_1, \ldots, e_n$ that span a vector space V, and they are linearly independent, then they are called a *basis* for V. It can be shown that any two bases for a vector space must have the same number of elements; this number is called the *dimension* of the vector space.

Given a basis $\{e_1, \ldots, e_n\}$ of a vector space, it can be proved that any vector x can be uniquely written as

$$x = \lambda_1 e_1 + \cdots + \lambda_n e_n \quad \text{for some scalars } \lambda_1, \ldots, \lambda_n.$$

The n-tuple $(\lambda_1, \ldots, \lambda_n)$ is called the *coordinates* of x with respect to the given basis. If we pick a different basis we'll get different coordinates, but of course the vector x itself is still the same regardless of the basis.

Inner products and orthogonality. Consider a vector space V over the field of real numbers. It is said to be an *inner product space* if the inner product satisfies these properties:

8. $v \cdot v \geq 0$ for all vectors v, and $v \cdot v = 0$ if and only if $v = 0$
9. $(\lambda u + \mu v) \cdot w = \lambda(u \cdot w) + \mu(v \cdot w)$ for all vectors u, v, w and scalars λ, μ
10. $v \cdot w = w \cdot v$ for all vectors v and w

An inner product space over the field of complex numbers is defined similarly, except that condition 10 is replaced by $v \cdot w = \overline{w \cdot v}$ where $\overline{\lambda}$ is the complex conjugate of the complex number λ. Also, the first part of condition 8 should be interpreted as $\mathrm{Im}(v \cdot v) = 0$ and $\mathrm{Re}(v \cdot v) \geq 0$.

Two vectors v and w in an inner product space are said to be *orthogonal* if $v \cdot w = 0$. A set of vectors (which may be finite or infinite) is said to be an *orthogonal system* if none of them is equal to 0 and in addition every pair of vectors in the set is orthogonal.

The *Euclidean norm* for an inner product space is

$$\|v\| = \sqrt{v \cdot v}.$$

A vector v with $\|v\| = 1$ is called a *unit vector*. An orthogonal system is said to be an *orthonormal system* if every vector in it is a unit vector.

5.1.2. USEFUL PROPERTIES

Here are some useful properties that can be proved from the abstract definitions.

11. $0v = 0$, for every vector v in a vector space.
12. $(-\lambda)v = -(\lambda v)$, for every vector v in a vector space and every scalar λ.
13. $(\lambda v) \cdot w = \lambda(v \cdot w)$, for all scalars λ and vectors v, w in an inner product space.
14. $0 \cdot v = 0$, for every vector v in an inner product space.
15. For all n and all scalars $\lambda_1, \ldots, \lambda_n$ and vectors $v_1, \ldots, v_n, w$ in an inner product space,

$$\Big(\sum_{i=1}^{n} \lambda_i v_i\Big) \cdot w = \sum_{i=1}^{n} \lambda_i (v_i \cdot w).$$

16. If $\{e_1, \ldots, e_n\}$ is an orthonormal system in an inner product space, then for every vector x in the span of $\{e_1, \ldots, e_n\}$, the coordinates of x are given by

$$x = \sum_{i=1}^{n} (x \cdot e_i)\, e_i.$$

17. $\|u + v\| \leq \|u\| + \|v\|$ for all vectors u, v; this is known as the *triangle inequality*.

These properties are mostly obvious when we're working with finite dimensional Euclidean space. For abstract vector spaces, they must be proved directly from the defining properties 1–10. The proofs are dull definition-pushing, but it's reassuring to know that it can be done. Here are some examples.

Exercise (Prove useful property 11). In this equation, the left hand side must be referring to the scalar $0 \in F$ and the right hand side to the vector $0 \in V$, where V is the vector space over field F, because otherwise the equation doesn't make sense—the abstract definitions don't define multiplication of vectors, and scalar multiplication yields a vector.

In both the real numbers and the complex numbers (and indeed in any field F), $0 = 0+0$. So, by property 6,

$$0v = (0 + 0)v = 0v + 0v.$$

By property 4, there is some vector $-(0v)$ such that $0v + \big(-(0v)\big) = 0$. Adding this to each side of the equation,

$$0v + \big(-(0v)\big) = \big(0v + 0v\big) + \big(-(0v)\big)$$

and so, using property 1,

$$0 = 0v + \big(0v + (-(0v))\big) = 0v + 0.$$

Finally, by property 3,

$$0 = 0v.$$

Exercise (Prove useful property 12). Property 6 says that

$$\lambda v + (-\lambda)v = \big(\lambda + (-\lambda)\big)v.$$

In both the real numbers and the complex numbers (and indeed in any field F), $\lambda + (-\lambda) = 0 \in F$, thus

$$\lambda v + (-\lambda)v = 0v$$

which we showed in the previous exercise to be equal to $0 \in V$. So $(-\lambda)v$ satisfies property 4 and it is therefore $-(\lambda v)$.

Exercise (Prove useful property 13).

$$\begin{aligned}(\lambda v) \cdot w &= \big((\lambda + 0)v\big) \cdot w \quad \text{since } \lambda = \lambda + 0 \in F \\ &= (\lambda v + 0v) \cdot w \quad \text{by property 6} \\ &= \lambda(v \cdot w) + 0(v \cdot w) \quad \text{by property 9} \\ &= \lambda(v \cdot w) \quad \text{since } 0\mu = 0 \in F.\end{aligned}$$

5.1.3. ADVANCED APPLICATION: FOURIER ANALYSIS

In this course on data science, the only vector space we're interested in is a simple finite-dimensional Euclidean space over the real numbers. Section 5.2 will go into detail. But first, to illustrate that there's some merit in defining vector spaces abstractly, here's an advanced application, a step on the way to Fourier analysis.

Inner product space. Let V consist of all continuous complex-valued functions on the interval $[-\pi, \pi]$. Define addition of functions in the obvious way, define multiplication by a complex number in the obvious way, and define the inner product to be

$$f \cdot g = \frac{1}{\pi} \int_{-\pi}^{\pi} f(\tau)\overline{g(\tau)}\, d\tau.$$

It is easy to check that properties 1–7 are satisfied, i.e. that this is a vector space over the field of complex numbers. Using some standard results about integration one can also show that properties 8–10 are also satisfied, therefore this is an inner product space. (A typical result: if f is a continuous function, then it is integrable over a finite interval.)

Orthonormal system. Every vector in V is a continuous function. Consider the vectors

$$\{e_1, e_2, \dots\} = \left\{ \frac{1}{\sqrt{2}},\ \cos(\tau),\ \sin(\tau),\ \cos(2\tau),\ \sin(2\tau),\ \cos(3\tau),\ \dots \right\}.$$

(The first element $1/\sqrt{2}$ is a way of writing the constant function $f(\tau) = 1/\sqrt{2}$.) With some A-level trigonometry and calculus, it can be shown that $e_i \cdot e_j = 0$ if $i \neq j$, and $e_i \cdot e_i = 1$ for every i, i.e. that this set is an orthonormal system.

Fourier coefficients. It can be shown that any function $f \in V$ can be written in coordinates of the orthonormal system as

$$f = \sum_{i=1}^{\infty} (f \cdot e_i)\, e_i$$

or equivalently

$$f(\tau) = \frac{a_0}{2} + \sum_{i=1}^{\infty} \Big(a_n \cos(n\tau) + b_n \sin(n\tau) \Big)$$

where

$$\begin{aligned} a_0 &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(\tau)\, d\tau, \\ a_i &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(\tau) \cos(i\tau)\, d\tau \quad \text{for } i \geq 1 \\ b_i &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(\tau) \sin(i\tau)\, d\tau \quad \text{for } i \geq 1. \end{aligned}$$

This goes beyond Useful Property 16: that property only applies to finite orthonormal systems, whereas here we have an infinite orthonormal system. In Part II *Computer Vision* you will see how the theory of inner product spaces is extended from finite to infinite bases.

5.2. Features in data

Key idea. A numerical feature can be seen as a vector, with one real number per object in the dataset. The vector space is over the field of real numbers. When we write a linear model like equation (23) on page 63, it should be interpreted as a linear combination of feature vectors.

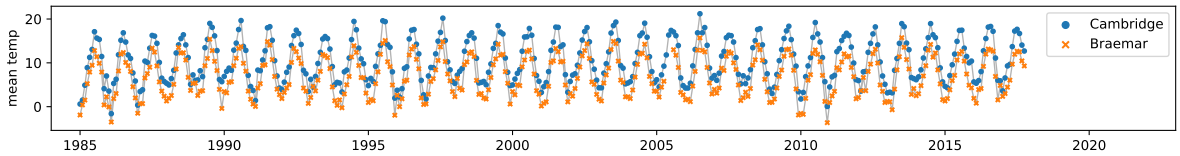
Feature vectors are a fundamental concept in machine learning. You will see them again in Part II *Machine Learning and Bayesian Inference*, *Natural Language Processing*, *Information Retrieval*, and anything at all to do with neural networks.

Let's illustrate with a dataset.

The UK Met Office makes available historic data[33] from 37 stations around the UK. Each station has monthly records for mean daily maximum temperature tmax, mean daily minimum temperature tmin, days of air frost af, total rainfall rain, and total sunshine duration sun. Coverage varies; the longest records are from Oxford and from Armagh, going back to 1853.

month	tmax	tmin	af	rain	sun	station	lat	lng	alt_m
1963 Sep	14.7	5.9	0	126.4	127.7	Eskdalemuir	55.311	-3.206	242
1955 Aug	–	–	–	35.1	194.7	Shawbury	52.794	-2.663	72
1937 May	15.3	8.4	0	59.8	184.8	Lowestoft	52.483	1.727	18
2007 Aug	20.6	11.8	0	40.3	204.6	Waddington	53.175	-0.522	68
1925 July	21.8	12.6	0	23.2	–	Sheffield	53.381	-1.490	131
⋮									

Here are two stations, Cambridge (measured at the National Institute of Agricultural Botany, between Churchill and Girton colleges), and Braemar in the Scottish highlands. The plot shows the mean temperature temp = (tmin + tmax)/2 as a function of date.



Are temperatures increasing? It's tricky to read this directly off the plot, because of the annual cycle and because of noise. A crude solution is to simply average over the 12 months of each year, and plot this average over time. This isn't ideal, because averaging is lossy i.e. we'd be throwing away data; and because a missing value for one month will cause the entire year to be missing.

A cleverer solution is to use features to model the effects we're trying to capture. There are two effects, an annual cycle, and a (hypothetical) increasing trend, which we can describe by the model

$$\mathsf{temp} \approx \alpha + \beta \sin(2\pi \mathsf{t} + \theta) + \gamma \mathsf{t}$$

where t is the date in years, and α, β, γ, and θ are unknown parameters. (The plot suggests that α is different for different stations, and the other parameters might also be different, so let's concentrate on a single station for now.)

Linear models are much easier to fit than non-linear models. The model we've proposed for tmean is linear in α and β and γ and not in θ—but there is a cunning trick from A-level trigonometry that lets us rewrite it as a linear model. The trick is

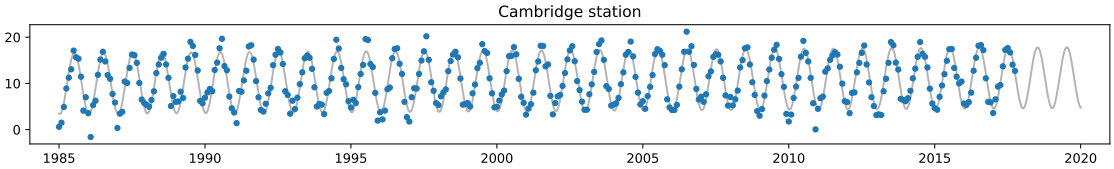
$$\sin(A + B) = \sin A \cos B + \cos A \sin B$$

and so our model can be rewritten

$$\mathsf{temp} \approx \alpha + \beta_1 \sin(2\pi \mathsf{t}) + \beta_2 \cos(2\pi \mathsf{t}) + \gamma \mathsf{t}.$$

[33] https://www.metoffice.gov.uk/public/weather/climate-historic

This model has four feature vectors: the constant vector 1; the vector s whose ith element is $s_i = \sin(2\pi t_i)$; the vector c whose ith element is $c_i = \cos(2\pi t_i)$; and the vector t. These are all n-dimensional vectors, where n is the number of rows in the dataset. There is also one outcome vector tmean. Here is the fitted model for Cambridge:



Why is α so extreme? It is the temperature in the year 1 BC (there was no year 0 AD), based on linearly extrapolating the rate γ. It's daft to trust that the model will predict well for such a wild extrapolation!

The parameters of the fitted model are $\alpha = -63.9$℃, $\beta_1 = -1.07$℃, $\beta_2 = -6.52$℃, and $\gamma = 0.0372$℃/year.

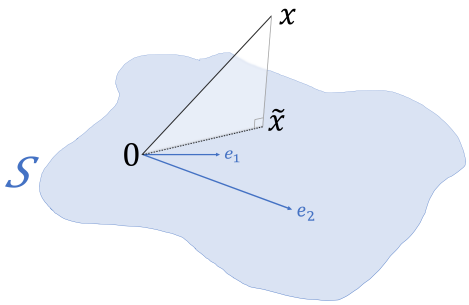
The model was fitted with similar sklearn.linear_model code to what we saw before:

```
# The data columns and features we'll use
t = data['yyyy'] + (data['mm']-1)/12
temp = (data['tmin']+data['tmax'])/2
one,s,c = np.ones(len(df)), np.sin(2*π*t), np.cos(2*π*t)
# We'll restrict attention to a subset of rows
station = data['station']
i = np.logical_and(station == 'Cambridge', t >= 1985)
# Fit a linear model (and tell it not to add its own constant '1' feature)
model = sklearn.linear_model.LinearRegression(fit_intercept=False)
model.fit(np.column_stack([one,s,c,t])[i,:], temp[i, np.newaxis])
print(model.coef_)
```

5.3. Orthogonal projection

Let's return to the key picture that illustrates linear mathematics.

- Linearly independent basis vectors e_1 and e_2
- The linear subspace spanned by those vectors, $S = \{\lambda_1 e_1 + \lambda_2 e_2 \,:\, -\infty < \lambda_1, \lambda_2 < \infty\}$
- Another vector x can be projected onto the subspace, by finding the point $\tilde{x} = \hat{\lambda}_1 e_1 + \hat{\lambda}_2 e_2$ in S that is closest to x
- The residual $x - \tilde{x}$ is orthogonal to the basis vectors



In Section 5.1 we reviewed vector spaces and bases. We'll now define projection in inner product spaces.

The Projection Theorem. Let V be an inner product space, let $\{e_1, \ldots, e_n\}$ be a finite collection of vectors, and let S be the subspace spanned by these vectors. Given a vector $x \in V$, there is a unique vector $\tilde{x}$ that is closest to x, i.e. that achieves

$$\|x - \tilde{x}\| = \min_{x' \in S} \|x - x'\|.$$

Mathematicians prefer to write inf rather than min in equations like this, where the minimum is being taken over an infinite set and it hasn't yet been established that the minimum is attained.

Furthermore, $x - \tilde{x}$ is orthogonal to S, i.e.

$$(x - \tilde{x}) \cdot y = 0 \quad \text{for all } y \in S.$$

The element $\tilde{x}$ is called the *orthogonal projection* of x onto S, and $x - \tilde{x}$ is called the *residual*.

Let's illustrate this theorem in three-dimensional Euclidean space. Let $e_1 = [1, 1, 0]$, let $e_2 = [1, 0, -1]$, and let $x = [1, 2, 3]$.

FINDING THE CLOSEST POINT

What is the closest point to x in the span of $\{e_1, e_2\}$? Just write out the optimization problem we want to solve:

$$\min_{\lambda_1, \lambda_2} \big\| x - (\lambda_1 e_1 + \lambda_2 e_2) \big\|.$$

We can compute the solution directly:

```
e1,e2,x = np.array([1,1,0]), np.array([1,0,-1]), np.array([1,2,3])
λ1,λ2 = scipy.optimize.fmin(lambda λ: np.linalg.norm(x-λ[0]*e1-λ[1]*e2), [0,0])
λ1*e1 + λ2*e2 # outputs: array([ 0.33332018,  2.66666169,  2.33334151])
```

Or we can try algebra. Expanding the definition of $\|\cdot\|$, we want to minimize

$$x \cdot x - 2\big(\lambda_1\, x \cdot e_1 + \lambda_2\, x \cdot e_2\big) + \big(\lambda_1^2\, e_1 \cdot e_1 + 2\lambda_1 \lambda_2\, e_1 \cdot e_2 + \lambda_2^2\, e_2 \cdot e_2\big).$$

Differentiating with respect to λ_1 and λ_2 and setting the derivatives equal to 0,

$$\begin{aligned} \frac{\partial}{\partial \lambda_1} = 0: \qquad & -2\, x \cdot e_1 + 2\lambda_1\, e_1 \cdot e_1 + 2\lambda_2\, e_1 \cdot e_2 = 0 \\ \frac{\partial}{\partial \lambda_2} = 0: \qquad & -2\, x \cdot e_2 + 2\lambda_1\, e_1 \cdot e_2 + 2\lambda_2\, e_2 \cdot e_2 = 0 \end{aligned} \tag{24}$$

or equivalently

$$\begin{aligned} \lambda_1\, e_1 \cdot e_1 + \lambda_2\, e_1 \cdot e_2 &= x \cdot e_1 \\ \lambda_1\, e_1 \cdot e_2 + \lambda_2\, e_2 \cdot e_2 &= x \cdot e_2. \end{aligned}$$

We can compute the solution to these equations:

```
λ1,λ2 = np.linalg.solve([[e1 @ e1, e1 @ e2], [e1 @ e2, e2 @ e2]],
                        [x @ e1, x @ e2])
λ1*e1 + λ2*e2 # outputs: array([ 0.33333333, 2.66666667, 2.33333333])
```

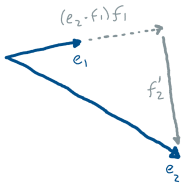
Or, for geometrical insight, we can rearrange equations (24) to get

$$\big(x - (\lambda_1 e_1 + \lambda_2 e_2)\big) \cdot e_1 = 0$$
$$\big(x - (\lambda_1 e_1 + \lambda_2 e_2)\big) \cdot e_2 = 0$$

In other words, the residual is orthogonal to e_1 and to e_2, and hence it's orthogonal to every linear combination of e_1 and e_2.

EXPLICIT PROJECTION VIA AN ORTHONORMAL BASIS

Another way to find $\tilde{x}$ is by creating an orthonormal basis out of $\{e_1, e_2\}$ and then applying Useful Property 16 on page 66 to get the coordinates of $\tilde{x}$. Let's create an orthonormal basis first. Start by setting f_1 to be a unit vector in the same direction as e_1:



$$f_1 = \frac{e_1}{\|e_1\|}.$$

Next, construct f_2 by subtracting the part that's parallel to f_1:

$$f_2' = e_2 - (e_2 \cdot f_1) f_1, \quad f_2 = \frac{f_2'}{\|f_2'\|}.$$

This construction ensures that $f_2' \cdot f_1 = 0$ therefore $f_2 \cdot f_1 = 0$, and it also ensures that both f_1 and f_2 are unit vectors. We've written f_1 and f_2 as linear combinations of e_1 and e_2, and it's easy to check that e_1 and e_2 can be written as linear combinations of f_1 and f_2, thus $\text{span}\{e_1, e_2\} = \text{span}\{f_1, f_2\} = S$.

Useful Property 16 now tells us exactly what the coordinates are for $\tilde{x}$:

$$\tilde{x} = (\tilde{x} \cdot f_1) f_1 + (\tilde{x} \cdot f_2) f_2.$$

Furthermore, the Projection Theorem tells us that the residual is orthogonal to $S = \text{span}\{f_1, f_2\}$, which means $(x - \tilde{x}) \cdot f_1 = (x - \tilde{x}) \cdot f_2 = 0$, thus

$$\tilde{x} = (x \cdot f_1) f_1 + (x \cdot f_2) f_2.$$

In numpy,

```
f1 = e1 / np.linalg.norm(e1)
f2' = e2 - (e2 @ f1) * f1
f2 = f2' / np.linalg.norm(f2')
(x@f1)*f1 + (x@f2)*f2 # outputs: array([ 0.33333333, 2.66666667, 2.33333333])
```

COLINEARITY

In this example, we projected onto linearly independent basis vectors e_1 and e_2. What happens if we want to project onto a collection of linearly dependent vectors, e.g. if $e_2 = \alpha e_1$?

The Projection Theorem doesn't assume linear independence, so the overall result still holds: there is still a unique projection $\tilde{x}$. The explicit projection method would still work, but it would give $f_2' = 0$, so we'd just discard that vector from the orthonormal basis. Equations (24) would still be correct, but they would have multiple solutions for λ_1 and λ_2.

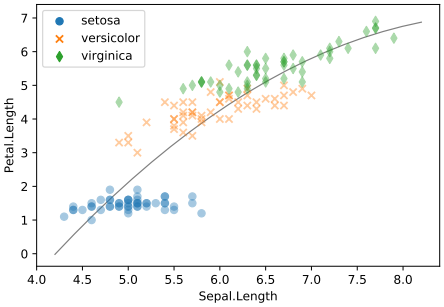
In other words, we can always project x onto a subspace $S = \text{span}(\{e_1, \ldots, e_n\})$, but we can only identify the 'contribution' of each of the e_i if the e_i are linearly independent.

5.4. Linear regression and least squares

Key idea. Suppose we want to fit a linear model to a dataset. If we model the outcome as a normal random variable, then maximum likelihood estimation of the unknown parameters is exactly the same as an orthogonal projection of the outcome vector onto feature vectors.

In the Iris dataset on page 63, we investigated how petal length depends on sepal length, and we proposed the model

$$\mathsf{Petal.Length} \approx \alpha + \beta\,\mathsf{Sepal.Length} + \gamma\,(\mathsf{Sepal.Length})^2.$$



Let's be explicit and propose a full probabilistic model:

$$\mathsf{Petal.Length}_i \sim \text{Normal}\Big(\alpha + \beta\,\mathsf{Sepal.Length}_i + \gamma\,(\mathsf{Sepal.Length}_i)^2,\ \sigma^2\Big) \tag{25}$$

where $i \in \{1, \dots, n\}$ indexes the rows of the dataset, and each $\mathsf{Petal.Length}_i$ is an independent random variable, and $\mathsf{Sepal.Length}_i$ is being treated as a non-random value.

We can estimate parameters in the usual way, by maximizing the likelihood of the observed values. For brevity, let $y_i = \mathsf{Petal.Length}_i$, let $e_i = \mathsf{Sepal.Length}_i$, and let $f_i = (\mathsf{Sepal.Length}_i)^2$. Then the density function for a single observation is

$$f(y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(y_i - (\alpha + \beta e_i + \gamma f_i)\right)^2 / 2\sigma^2}$$

and the log likelihood is

$$\log \text{lik}(\alpha, \beta, \gamma, \sigma \mid y) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\Big(y_i - (\alpha + \beta e_i + \gamma f_i)\Big)^2.$$

We can maximize the log likelihood in two steps. The first step is to maximize the last term, i.e. find $\hat{\alpha}$, $\hat{\beta}$, and $\hat{\gamma}$ that solve

$$\min_{\alpha,\beta,\gamma} \big\|y - (\alpha 1 + \beta e + \gamma f)\big\|^2.$$

In this equation we have switched to vector notation, and 1 means the vector $[1, 1, \dots, 1]$. This is nothing other than finding the orthogonal projection of the outcome vector y onto the space spanned by the feature vectors $\{1, e, f\}$. Another name for this procedure is the *method of least squares*, invented by Gauss.

The second step is to find σ to maximize what's left, i.e. to solve

$$\min_{\sigma > 0}\left\{\frac{n}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\big\|y - (\hat{\alpha}1 + \hat{\beta}e + \hat{\gamma}f)\big\|^2\right\}.$$

This is a trivial one-parameter optimization problem, once we know $\hat{\alpha}$, $\hat{\beta}$, and $\hat{\gamma}$.

CONFOUNDED FEATURES

The Projection Theorem says that there is a unique projection $\tilde{y}$ of y onto the features $\{1, e, f\}$. However, if the feature vectors were linearly dependent, then there would not be a unique solution for (α, β, γ), and we'd say that the features are *confounded*.

In the context of abstract vector spaces, the word is *colinearity*; see Section 5.3. Sometimes, geometrical intuition about colinearity can help us debug what's going wrong with a probabilistic model fit.

Example sheet 3b asks you to look at examples of confounding.

INFERENCE

An explicit probabilistic model like equation (25) lets us make inferences, using the techniques from Section 3 and Example Sheet 2.

- We can use Bayesian reasoning: invent prior distributions for the unknown parameters, and calculate their posterior distributions.
- We can use frequentist reasoning: compute confidence intervals for the unknown parameters, using bootstrap resampling. Resampling means generating synthetic datasets based on the data we actually saw; the natural resampling method here is to compute the maximum likelihood estimates for the parameters, and then to plug these estimates into the model (25) and generate new random variables. This is known as *parametric resampling*.
- We can conduct hypothesis tests, along the lines of example sheet 2 question 3.

Example sheet 3b asks you to look at all these types of inference.

5.5. Feature engineering

In Section 5.2, we cleverly designed features to allow us to extract an underlying linear trend from climate data, taking account of the annual cycle. In general, we design features for several purposes:

- features to extract a particular summary from the data, e.g. the linear trend in the climate data;
- features that correspond to a causal model for which we want to estimate parameters, e.g. the transition probabilities for a Markov chain;
- 'black box' features that capture enough detail for us to be able to make good predictions or extrapolations—we don't have to understand such features, we just want them to work well;
- features that turn arbitrary objects like tweets or sentence fragments into numbers that can be put into quantitative models, e.g. distributional semantics which you will study in Part II *Natural Language Processing*, and term frequency models for documents which you will study in Part II *Information Retrieval*.

Example sheet 3b invites you to use features for one-hot coding of categorical data, and for time series analysis.

* * *

The more features we add, the better the fit i.e. the smaller the residual we can achieve. But models with too many features tend to be bad at generalizing to new data (see example sheet 2 question 5). It's an art to design sets of features that are expressive enough to capture the meaningul variation in the data, while being parsimonious enough to generalize well. Here are two strategies that are sometimes helpful. You will learn more about them in futher courses on machine learning and data science.

Feature selection. Start with a long list of possible features. Pick m, a number of features to use, and find the best fitting model subject to the constraint that it's only allowed to use m of the possible features. This is called *feature selection*.

Dimension reduction. Start with a long list of possible features $\{e_1, \ldots, e_n\}$. Pick m, a number of features to use, and construct a set of m vectors $\{f_1, \ldots, f_m\}$ that capture the features as well as possible. For example, we might set $m = 2$ and pick $\{f_1, f_2\}$ to minimize $\sum_{i=1}^{n} \|e_i - \tilde{e}_i\|$, where $\tilde{e}_i$ is the projection of vector e_i onto the span of $\{f_1, f_2\}$. This would be called a *two-dimensional embedding* of the features, and it is an example of *dimension reduction*.

Gauss' invention of the Method of Least Squares

The method of least squares is the automobile of modern statistical analysis: despite its limitations, occasional accidents, and incidental pollution, it and its numerous variations, extensions, and related conveyances carry the bulk of statistical analyses, and are known and valued by nearly all.

Stephen M. Stigler, *Gauss and the invention of least squares*, Annals of Statistics, 1981

Gauss gives an insightful and illuminating account of how the idea of the least-squares method came to him. Up to that time, "… in every case in which it was necessary to deduce the orbits of heavenly bodies from observations, there existed advantages not to be despised, suggesting, or at any rate permitting, the application of special methods; of which advantages the chief one was, that by means of hypothetical assumptions an approximate knowledge of some elements could be obtained before the computation of the elliptic elements was commenced. Notwithstanding this, it seems somewhat strange that the general problem – To determine the orbit of a heavenly body, without any hypothetical assumption, from observations not embracing a great period of time, and not allowing the selection with a view to the application of special methods, – was almost wholly neglected up to the beginning of the present century; or at least, not treated by any one in a manner worthy its importance; since it assuredly commended itself to mathematicians by its difficulty and elegance, even if its great utility in practice were not apparent. An opinion had universally prevailed that a complete determination from observations embracing a short interval of time was impossible – an ill-founded opinion – for it is now clearly shown that the orbit of a heavenly body may be determined quite nearly from good observations embracing only a few days; and this without any hypothetical assumption.

“Some idea occurred to me in the month of September of the year 1801, engaged at the time on a very different subject, which seemed to point to the solution of the great problem of which I have spoken. Under such circumstances we not unfrequently, for fear of being too much led away by an attractive investigation, suffer the associations of ideas, which more attentively considered, might have proved most fruitful in results, to be lost from neglect. And the same fate might have befallen these conceptions, had they not happily occurred at the most propitious moment for their preservation and encouragement that could have been selected. For just about this time the report of the new planet, discovered on the first day of January of that year with the telescope at Palermo, was the subject of universal conversation; and soon afterwards the observations made by the distinguished astronomer Piazzi from the above date to the eleventh of February were published. Nowhere in the annals of astronomy do we meet with so great an opportunity, and a greater one could hardly be imagined, for showing most strikingly, the value of this problem, than in this crisis and urgent necessity, when all hope of discovering in the heavens this planetary atom, among innumerable small stars after the lapse of nearly a year, rested solely upon a sufficiently approximate knowledge of its orbit to be based upon these very few observations. Could I ever have found a more seasonable opportunity to test the practical value of my conceptions, than now in employing them for the determination of the orbit of the planet Ceres, which during the forty-one days had described a geocentric arc of only three degrees, and after the lapse of a year must be looked for in a region of the heavens very remote from that in which it was last seen? This first application of the method was made in the month of October, 1801, and the first clear night, when the planet was sought for (by de Zach, December 7, 1801) as directed by the numbers deduced from it, restored the fugitive to observation. Three other new planets, subsequently discovered, furnished new opportunities for examining and verifying the efficiency and generality of the method.

“Several astronomers wished me to publish the methods employed in these calculations immediately after the second discovery of Ceres; but many things – other occupations, the desire of treating the subject more fully at some subsequent period, and, especially, the hope that a further prosecution of this investigation would raise various parts of the solution to a greater degree of generality, simplicity, and elegance, – prevented my complying at the time with these friendly solicitations. I was not disappointed in this expectation, and I have no cause to regret the delay. For the methods first employed have undergone so many and such great changes, that scarcely any trace of resemblance remain between the method in which the orbit of Ceres was first computed, and the form given in this work. Although it would be foreign to my purpose, to narrate in detail all the steps by which these investigations have been gradually perfected, still, in several instances, particularly when the problem was one of more importance than usual, I have thought that the earlier methods ought not to be wholly suppressed. But in this work, besides the solution of the principal problems, I have given many things which, during the long time I have been engaged upon the motions of the heavenly bodies in conic sections, struck me as worthy of attention, either on account of their analytical elegance, or more especially on account of their practical utility.”

Quirino Paris, The dual of the least-squares method, 2014.

Quoting Gauss, Theoria motus corporum coelestium in sectionibus conicis solem ambientum, 1809.

Translation by Davis, Theory of the motion of the heavenly bodies moving about the sun in conic sections, 1857.