# 2. Feature spaces / linear regression

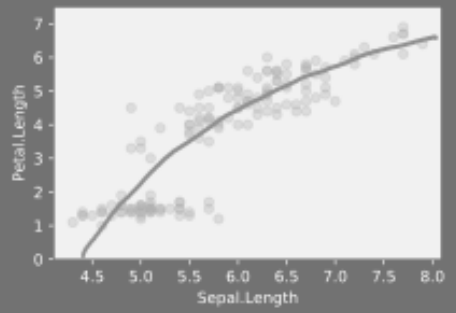
# 2.3. Linear mathematics

Exercise 2.1.

The Iris dataset, popularized by Ronald Fisher (a genius who almost single-handedly created the foundations for modern statistical science), has 50 records of iris measurements, from three species.

| 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 |
| 4.2 | 1.2 | 5.7 | 3.0 | versicolor |
| ... | | | | |

How does `Petal.Length` depend on `Sepal.Length`?




Let's guess that for parameters $\alpha, \beta, \gamma$ (to be estimated),

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



This is called a *linear model* because it can be written in *linear algebra* form, using vectors for the entire dataset.

The response vector is

$\texttt{Petal.Length} = [\mathrm{PL}_1, \mathrm{PL}_2, \ldots, \mathrm{PL}_n]$

The feature vectors are

$\text{one} = [1, 1, \ldots, 1]$

$\texttt{Sepal.Length} = [\mathrm{SL}_1, \mathrm{SL}_2, \ldots, \mathrm{SL}_n]$

$(\texttt{Sepal.Length})^2 = [(\mathrm{SL}_1)^2, (\mathrm{SL}_2)^2, \ldots, (\mathrm{SL}_n)^2]$

The response vector is predicted by a linear combination of feature vectors:

$$\begin{bmatrix} \mathrm{PL}_1 \\ \mathrm{PL}_2 \\ \vdots \\ \mathrm{PL}_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} \mathrm{SL}_1 \\ \mathrm{SL}_2 \\ \vdots \\ \mathrm{SL}_n \end{bmatrix} + \gamma \begin{bmatrix} (\mathrm{SL}_1)^2 \\ (\mathrm{SL}_2)^2 \\ \vdots \\ (\mathrm{SL}_2)^2 \end{bmatrix}$$

Vector equation

↓

Linear Algebra

## NST Maths A, Michaelmas

### 1.7.2 Shortest distance of a point from a plane

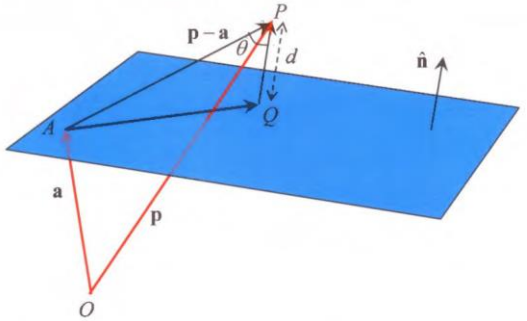



Figure 13: The shortest distance between the point $P$ and the plane defined by the point $A$ and normal $\hat{\mathbf{n}}$.

Consider the plane that passes through point $A$ (given by position vector $\mathbf{a}$) and that has unit normal $\hat{\mathbf{n}}$. From (11), the equation for the plane is defined by the points $\mathbf{r}$ satisfying

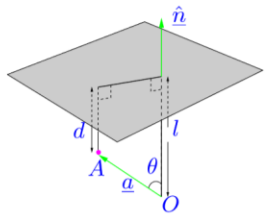
$$(\mathbf{r}-\mathbf{a})\cdot\hat{\mathbf{n}}=0\,.$$

The closest point on the plane to a point $P$, given by position vector $\mathbf{p}$, is the point $Q$, where $\overrightarrow{QP}$ is normal to the plane and $\left|\overrightarrow{QP}\right|=d$.

## NST Maths B, Michaelmas

### Example: Distance of point from plane

- What is distance of point $A$ with position vector $\underline{a}$ from plane $\underline{r}\cdot\hat{\underline{n}}=l$?




- Line containing $A$ and point of closest approach of plane to $A$ must be $\parallel\hat{\underline{n}}$; has equation

$$\underline{r}=\underline{a}+\lambda\hat{\underline{n}}$$

- Line meets plane where $\underline{r}\cdot\hat{\underline{n}}=l$, i.e. where

$$l=\underline{a}\cdot\hat{\underline{n}}+\lambda$$

- $\lambda$ is distance along line from $\underline{a}$ so required distance is $|\underline{a}\cdot\hat{\underline{n}}-l|$

## NST Maths A, Easter

### Orthogonality - 2/3

- The **vectors** $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ are a **basis** of **orthonormal vectors** in $\mathbb{R}^3$:

$$\mathbf{e}_i\cdot\mathbf{e}_j=\delta_{ij}\,. \qquad (2)$$

- We use the **orthogonality properties** (2) to **calculate** the components of $\mathbf{a}$:

$$\mathbf{e}_1\cdot\mathbf{a}=a_1\times1+a_2\times0+a_3\times0=a_1\,.$$

In general

$$a_i=\mathbf{e}_i\cdot\mathbf{a}\,,\quad \text{for}\quad i=1,2,3,. \qquad (3)$$

- The above **generalises** to Euclidean space $\mathbb{R}^n$ with

$$\mathbf{a}=a_i\,\mathbf{e}_i\,,\quad \text{for}\quad i=1,\ldots,n,.$$

the **components** $a_i$ are **evaluated** in the **same way** as in the case with $n=3$ because (2) and (3) **still hold**, but with $i\,,j$ now in the **range** 1 to $n$.

## NST Maths B, Easter

*Definition.* $V$ is called a **vector space over** $K$, and the elements of $V$ are called **vectors**, if the following **axioms** hold:

**A1** For any vectors $u,v,w\in V$, $(u+v)+w=u+(v+w)$. **(Associativity.)**

**A2** For any vectors $u,v\in V$, $u+v=v+u$. **(Commutativity.)**

**A3** There is a vector in $V$ denoted $0$, called the **zero vector** for which $u+0=u$ $\forall\,u\in V$.

**A4** For each vector $u\in V$ there is a vector in $V$ denoted $-u$ for which $u+(-u)=0$. **(Inverse.)**

**A5** For any $a\in K$ and any $u,v\in V$, $a(u+v)=au+av$.

**A6** For any $a,b\in K$ and any $u\in V$, $(a+b)u=au+bu$.

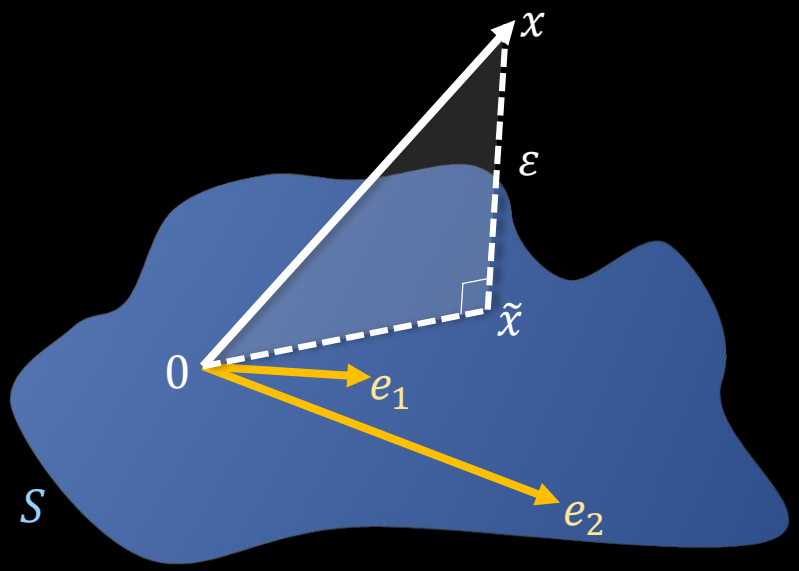
**A7** For any $a,b\in K$ and any $u\in V$, $(ab)u=a(bu)$.

**A8** For the unit scalar $1\in K$ and any $u\in V$, $1u=u$.

In the lecture notes appendix, you can find

- an axiomatic introduction to linear algebra
- an brief sketch of the linear algebra of functions ⇒ Fourier analysis

# The key ideas from linear mathematics




"The subspace $S$ spanned by $e_1$ and $e_2$."

- The subspace spanned by $\{e_1, \dots, e_K\}$ is the set of all linear combinations

$$S = \{\lambda_1 e_1 + \cdots + \lambda_K e_K : \lambda_k \in \mathbb{R} \text{ for all } k\}$$

"Linearly independent vectors $e_1$ and $e_2$."

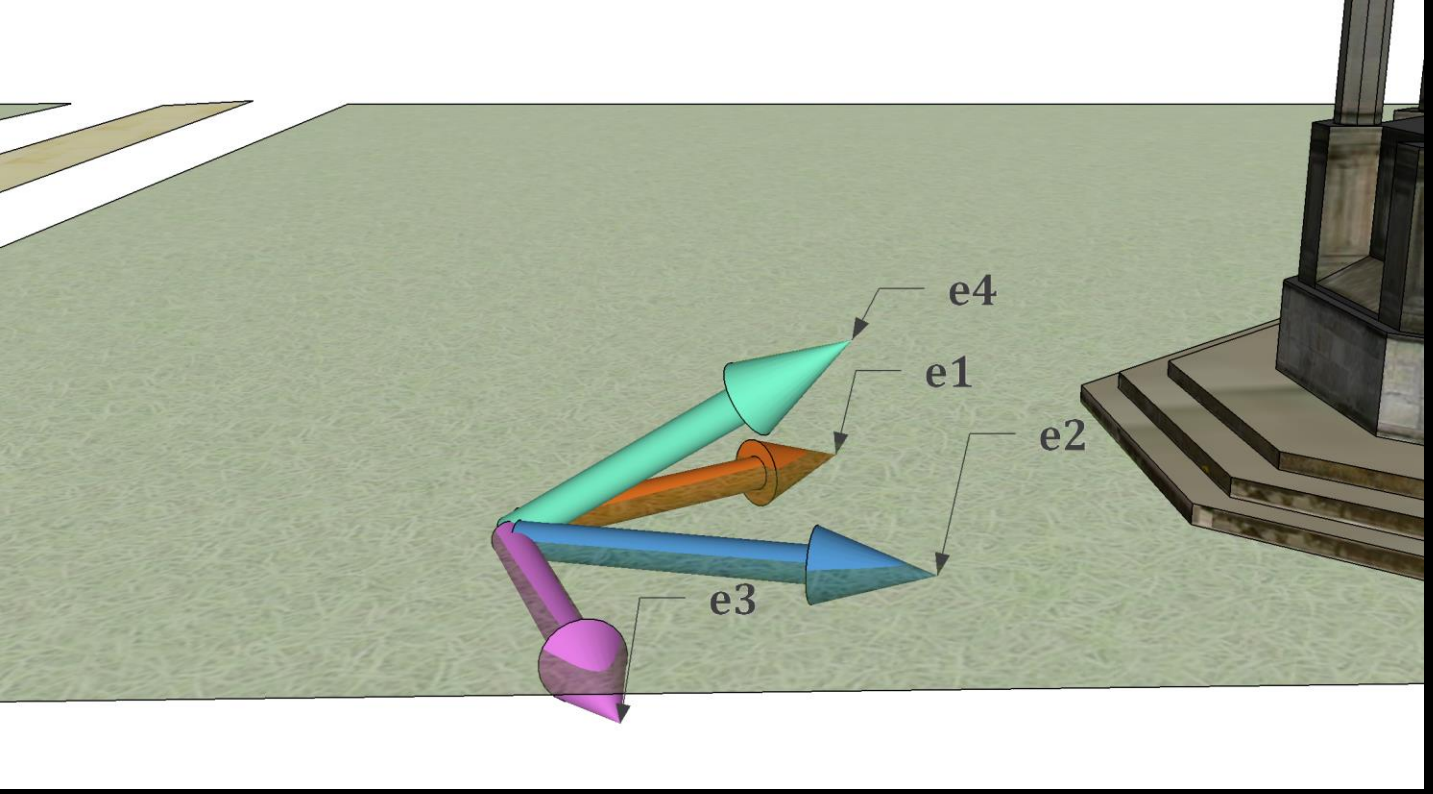
- A collection of vectors $\{e_1, \dots, e_K\}$ is linearly dependent if there is some set of weights $\{\lambda_1, \dots, \lambda_K\}$, not all 0, such that

$$\lambda_1 e_1 + \cdots + \lambda_K e_K = 0$$

  If so, then at least one of the $e_i$ can be written as a linear combination of the others.

- If not, then the vectors are linearly independent.

- $$\text{rank}\left(\begin{bmatrix} | & & | \\ e_1 & \cdots & e_K \\ | & & | \end{bmatrix}\right) \text{ is } \begin{cases} = K & \text{if linearly independent} \\ < K & \text{if linearly dependent} \end{cases}$$




$\{e_1, e_2, e_3\}$ are linearly dependent — it looks like $e_2 = e_1 + e_3$

$\Leftrightarrow \quad e_1 - e_2 + e_3 = 0.$

To obtain a linearly independent set,
I could simply discard any one out of $e_1, e_2, e_3$.

For example, $\{e_1, e_2, e_4\}$ is a linearly independent set. whose span is $\mathbb{R}^3$.

Exercise.

Are the following three vectors linearly independent? If not, find a subset that is.

$$e_1 = [1,1,1,1]$$
$$e_2 = [0,1,1,0]$$
$$e_3 = [1,0,0,1]$$

Not LI.

$e_1 = e_2 + e_3$

```
1   e1,e2,e3 = [1,1,1,1], [0,1,1,0], [1,0,0,1]
2   numpy.linalg.matrix_rank(numpy.column_stack([e1,e2,e3]))  # returns 2
```

To get a LI set:

$\{e_1, e_3\}$

$\{e_1, e_2\}$

$\{e_2, e_3\}$

$\{e_1 + e_3, e_3\}$

These are all LI sets that span the same space.

Exercise.

Are the following five vectors linearly independent? If not, find a subset that is.

$e_1 = [1,1,1,1]$
$e_2 = [0,1,1,0]$
$e_3 = [1,0,0,1]$
$e_4 = [1,1,1,0]$
$e_5 = [0,0,0,1]$

**Linearly dependent**: there are two linear relations between the vectors.

$e_2 + e_3 = e_4 + e_5$ –

$e_2 + e_3 = e_1$

Some LI subsets: $\{e_1, e_2, e_4\}$ or $\{e_1, e_3, e_5\}$ or ...

Are $\{e_1, e_2, e_4\}$ linearly independent?

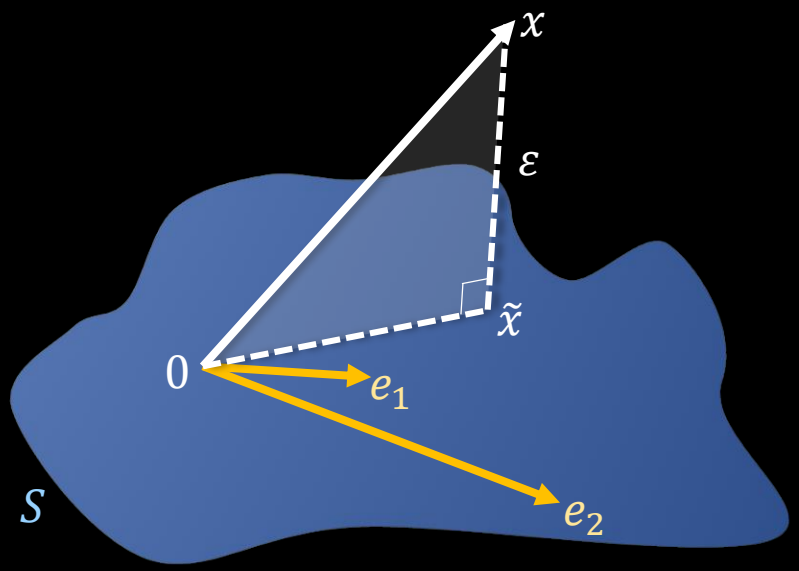
suppose $\lambda_1 e_1 + \lambda_2 e_2 + \lambda_4 e_4 = 0$

$$\Rightarrow \lambda_1 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \lambda_4 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = 0$$

$$\Rightarrow \left.\begin{aligned} \lambda_1 + \lambda_4 &= 0 \\ \lambda_1 + \lambda_2 + \lambda_4 &= 0 \\ \lambda_1 + \lambda_2 + \lambda_4 &= 0 \\ \lambda_1 &= 0 \end{aligned}\right\} \Rightarrow \lambda_4 = 0 \Rightarrow \lambda_2 = 0$$

So, linearly independent.

# The key ideas from linear mathematics




"The projection of $x$ onto $\tilde{x} \in S$."

- Given a space $S$ spanned by $\{e_1, \dots, e_K\}$, and another vector $x$, there is a unique closest point in $S$, i.e. a unique vector $\tilde{x}$ that solves

$$\tilde{x} = \arg\min_{y \in S} \|x - y\|^2$$

- Since $\tilde{x} \in S$ it can be written as a linear combination

$$\tilde{x} = \hat{\lambda}_1 e_1 + \cdots + \hat{\lambda}_K e_K$$

If the $e_i$ are linearly independent, there is a unique solution for the $\hat{\lambda}_i$. Otherwise there are many ways to write the linear combination.

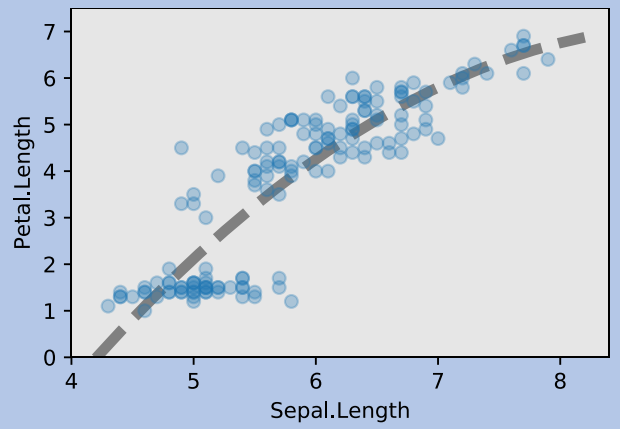
- The residual $\varepsilon = x - \tilde{x}$ is orthogonal to $S$, i.e. $\varepsilon \cdot y = 0$ for all $y \in S$. This lends $\tilde{x}$ the name "orthogonal projection of $x$ onto $S$".

A *linear model* is a model of the form

$$y = \beta_1 e_1 + \cdots + \beta_K e_K + \varepsilon$$

- $y$ is the response vector $[y_1, y_2, \ldots, y_n]$
- $e_1, \ldots, e_K$ are features, each a vector of length $n$
- $\varepsilon = [\varepsilon_1, \ldots, \varepsilon_n]$ is the *residuals* vector / error / noise
- After fitting the model the *predicted values* are

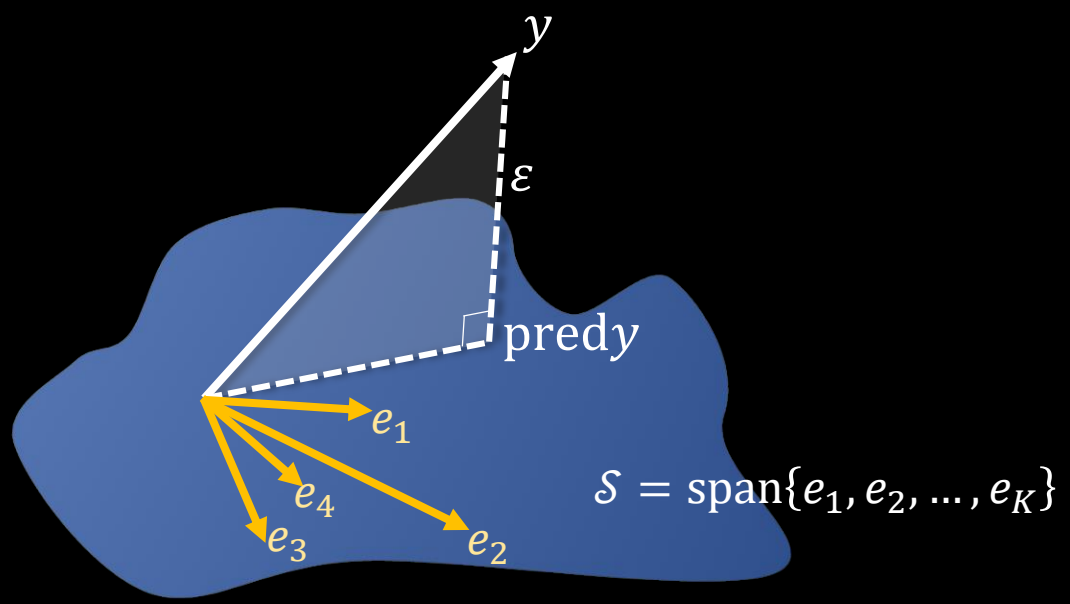
$$\text{pred}y = \hat{\beta}_1 e_1 + \cdots + \hat{\beta}_K e_K$$



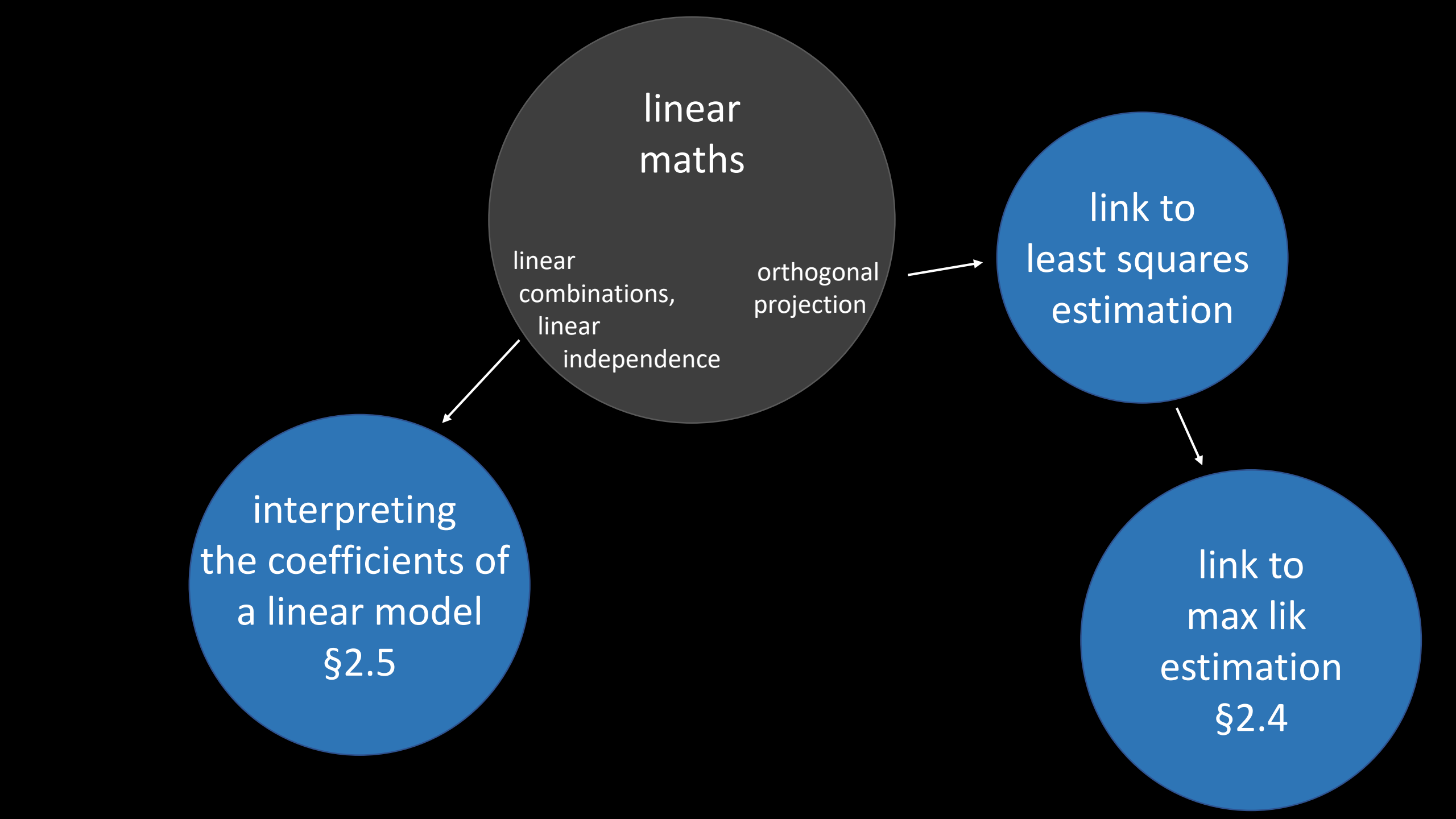

A sensible way to fit a linear model is to pick the $\beta$ coefficients so as to minimize the *mean square error*

$$\text{mse} = \frac{1}{n}\sum_{i=1}^{n} \varepsilon_i^2$$

$$= \frac{1}{n}\|y - \text{pred}y\|^2$$



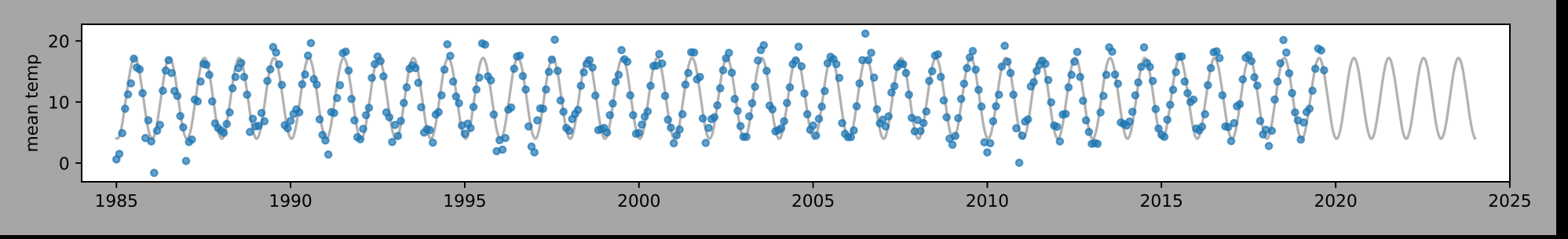

$\mathcal{S}$ is called the *feature space* of the model. The bigger the feature space, the more responses the model can describe.


linear maths
linear combinations, linear independence
orthogonal projection
link to least squares estimation
interpreting the coefficients of a linear model §2.5
link to max lik estimation §2.4

# 2.5 Interpreting a linear model's coefficients

To interpret the coefficients from a linear model,

1. Write out the predicted response for a generic datapoint.
   This helps you see what the coefficients mean.
2. Write out the feature vectors and ask:
   "are they linearly dependent?"
   If they are, the coefficients have no intrinsic meaning.




Model A: $\texttt{temp} \approx \alpha\,\texttt{1} + \beta_1 \sin(2\pi\,\texttt{t}) + \beta_2 \cos(2\pi\,\texttt{t}) + \gamma\,\texttt{t}$ $\Rightarrow \hat{\alpha} = -60.5$

Model B: $\texttt{temp} \approx \alpha\,\texttt{1} + \beta_1 \sin(2\pi\,\texttt{t}) + \beta_2 \cos(2\pi\,\texttt{t}) + \gamma(\texttt{t} - 2000)$ $\Rightarrow \hat{\alpha} = 10.6$

To figure out what the coefs mean, let's write out predictions for a "generic" datapoint.

Model A:
pred. temp at $t=0$ (Jan in 0AD) is $\alpha + \beta_1 \sin(2\pi 0) + \beta_2 \cos(2\pi 0) + \gamma 0 = \alpha + \beta_2$

Model B:
pred. temp at $t=0$ : $\alpha + \beta_2 - 2000\gamma$
pred temp at $t=2000$ : $\alpha + \beta_2$

In model A, $\hat{\alpha}$ reports the fitted temp in Jan 0AD
In model B, $\hat{\alpha}$ reports the fitted temp in in Jan 2000
} after removing the annual cycle

Model A: $\texttt{temp} \approx \alpha\,\texttt{1} + \beta_1 \sin(2\pi\,\texttt{t}) + \beta_2 \cos(2\pi\,\texttt{t}) + \gamma\,\texttt{t}$ $= \alpha\,\underline{1} + \beta_1\,\underline{s} + \beta_2\,\underline{c} + \gamma\,\underline{t}$

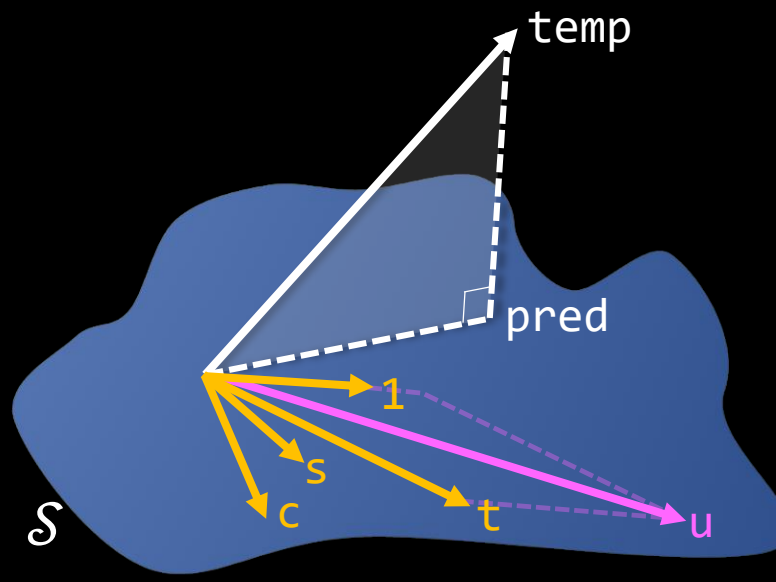
Model B: $\texttt{temp} \approx \alpha\,\texttt{1} + \beta_1 \sin(2\pi\,\texttt{t}) + \beta_2 \cos(2\pi\,\texttt{t}) + \gamma(\texttt{t} - 2000)$ $= \alpha\,\underline{1} + \beta_1\,\underline{s} + \beta_2\,\underline{c} + \gamma\,\underline{u}$

- The feature space is identical since $\underline{u} = \underline{t} - 1000\,\underline{1}$ = a linear combination of $\underline{t}$ and $\underline{1}$

$$\mathcal{S} = \text{span}\{\texttt{1}, \texttt{s}, \texttt{c}, \texttt{t}\} = \text{span}\{\texttt{1}, \texttt{s}, \texttt{c}, \texttt{u}\}$$

- Thus, when we fit the model (project `temp` onto $\mathcal{S}$ to get `pred`) we'll obtain the same predictions in each case.

- The two models are just different coordinate systems for $\mathcal{S}$. Their fitted coefficients are coordinates for `pred`.



code: Azure notebook

Exercise 2.4 (Confounded features).

The UK Home Office makes available a dataset of police stop-and-search incidents. We wish to investigate whether there is racial bias in police decisions to stop-and-search. Using the binary response vector

$$y = 1[\texttt{outcome} \neq \text{"False"}]$$

and the linear model

$$y \approx \alpha + \beta_{\texttt{eth}}$$

analyse whether there is racial bias in policing actions.

| force | datetime | object of search | location_ street_name | gender | age_range | officer_defined_ ethnicity | outcome |
|---|---|---|---|---|---|---|---|
| metropolitan | 2017-09-11 T16:35:00 | Controlled drugs | On or near Supermarket | Female | 10-17 | White | False |
| metropolitan | 2018-11-09 T01:20:00 | Anything to threaten or harm anyone | On or near Penge Road | Male | 18-24 | Black | Community resolution |
| thames-valley | 2019-06-25 T13:10:00 | Controlled drugs | NaN | Male | 10-17 | White | A no further action disposal |
| kent | 2019-04-13 T15:14:00 | Evidence of offences under the Act | On or near Waterlands Lane | Female | 25-34 | Other | A no further action disposal |

code: Azure notebook

Exercise 2.4 (Confounded features).
The UK Home Office makes available a dataset of police stop-and-search incidents. We wish to investigate whether there is racial bias in police decisions to stop-and-search. Using the binary response vector

$$y = 1[\texttt{outcome} \neq \text{"False"}]$$

and the linear model

$$y \approx \alpha + \beta_{\texttt{eth}}$$

analyse whether there is racial bias in policing actions.

$y_i = \begin{cases} 1 & \text{if record } i \text{ has outcome} \neq \text{"False"}; \text{ ie if the police find something} \\ 0 & \text{——————} = \text{"False"} \quad \text{ie if the police find nothing.} \end{cases}$

$\text{mean of } y = \frac{\#\text{cases where } y_i = 1}{\text{total } \#\text{records}} = \frac{\#\text{stops where police find sthg}}{\#\text{stops}} = \mathbb{P}(\text{police find sthg})$

## How can we use this model to look for racial bias?

Suppose for example $\beta_{\text{"Black"}}$ is low, compared to other groups

$\Rightarrow$ the predicted value of $y$ for an $\texttt{eth} = \text{"Black"}$ person, i.e. $\alpha + \beta_{\text{Black}}$, is low

$\Rightarrow$ the probability of the police finding something is low, for an $\texttt{eth} = \text{"Black"}$ person

$\Rightarrow$ the police are stopping relatively more innocent people of $\texttt{eth} = \text{"Black"}$

$\Rightarrow$ the police are biased against $\texttt{eth} = \text{"Black"}$.

So, we can read off the $\beta$ coefficients to learn about racial bias.

```
police.groupby('outcome').apply(len)
print('Missing values:', numpy.sum(pandas.isnull(police['outcome'])))
```

```
outcome
A no further action disposal                                467106
Arrest                                                       93384
Article found - Detailed outcome unavailable                  6705
Caution (simple or conditional)                               2947
Community resolution                                         35319
False                                                       239660
Khat or Cannabis warning                                     18668
Local resolution                                              8164
Offender cautioned                                            1747
Offender given drugs possession warning                      26030
Offender given penalty notice                                 5623
Penalty Notice for Disorder                                  10628
Summons / charged by post                                    11168
Suspect arrested                                             63191
Suspect summonsed to court                                    5758
Suspected psychoactive substances seized - No further action    17
Missing values: 17800
```

Bothersomely, pandas.groupby won't warn you about missing values. They're easy to forget about!

code: Azure notebook

```
police.groupby('officer_defined_ethnicity').apply(len)
print('Missing values:', numpy.sum(pandas.isnull(police['officer_defined_ethnicity'])))
```

```
officer_defined_ethnicity
Asian    125646
Black    253315
Mixed      1644
Other     27809
White    532584
Missing values: 72917
```

code: Azure notebook

```
# Remove rows with missing values
# Note: it is good practice to treat your source data as immutable,
# and to use generic variable names like "df" for derived objects of local scope.
bad = pandas.isnull(police['outcome']) | pandas.isnull(police['officer_defined_ethnicity'])
df = police.loc[~bad].copy()

# Tidy up -- give the features of interest shorter names
df['y'] = numpy.where(df['outcome'] != 'False', 1, 0)
df['eth'] = df['officer_defined_ethnicity']

# Eyeball
df.groupby(['y','eth']).apply(len).unstack()
```

| eth<br>y | Asian | Black | Mixed | Other | White |
|---|---|---|---|---|---|
| 0 | 27029 | 59917 | 134 | 6387 | 128370 |
| 1 | 97019 | 192269 | 1483 | 21162 | 394863 |

Fitting the linear model $y \approx \alpha + \beta_{\text{eth}}$ using one-hot coding to extract the $\beta$ coefficients:

code: Azure notebook

```
ethnicity_levels = numpy.unique(df['eth'])
eth_onehot = [df['eth']==i for i in ethnicity_levels]

model = sklearn.linear_model.LinearRegression()
model.fit(numpy.column_stack(eth_onehot), df['y'])
α,βs = model.intercept_, model.coef_

print(f'α = {α}')
for i,β in zip(ethnicity_levels, βs):
    print(f'β[{i}] = {β}')
```

$e_{As} = 1[\text{eth} = \text{"Asian"}]$
$e_{Bl} = 1[\text{eth} = \text{"Black"}]$
etc.

```
α = 171497104.17042407
β[Asian] = -171497103.3887985
β[Black] = -171497103.40795302
β[Mixed] = -171497103.2539072
β[Other] = -171497103.40197015
β[White] = -171497103.41569293
```

These coefficients look like nonsense!

What is $\alpha$ meant to represent?

Is it if meant to tell us about mean value of $y$ (from the model — $y \approx \alpha + \beta_{\text{eth}}$)

ie is it if supposed to be a probability?

# Linear dependence among feature vectors

To gain insight into what these reported coefficients mean, let's look at the feature vectors. We fitted the model $y \approx \alpha + \beta_{\texttt{eth}}$, using one-hot coding. More explicitly,

$$y \approx \alpha\, 1 + \beta_{\text{As}} e_{\text{As}} + \beta_{\text{Bl}}\, e_{\text{Bl}} + \beta_{\text{Mi}}\, e_{\text{Mi}} + \beta_{\text{Oth}}\, e_{\text{Oth}} + \beta_{\text{Wh}}\, e_{\text{Wh}}$$

These features are linearly dependent.
Therefore, any fitted model can be rewritten with different coefficients:

$$\hat{\alpha}\, 1 + \hat{\beta}_{\text{As}} e_{\text{As}} + \hat{\beta}_{\text{Bl}} e_{\text{Bl}} + \hat{\beta}_{\text{Mi}} e_{\text{Mi}} + \hat{\beta}_{\text{Oth}} e_{\text{Oth}} + \hat{\beta}_{\text{Wh}} e_{\text{Wh}}$$
$$\approx (\hat{\alpha} + \hat{\beta}_{\text{As}}) e_{\text{As}} + (\hat{\alpha} + \hat{\beta}_{\text{Bl}}) e_{\text{Bl}} + (\hat{\alpha} + \hat{\beta}_{\text{Mi}}) e_{\text{Mi}} + (\hat{\alpha} + \hat{\beta}_{\text{Oth}}) e_{\text{Oth}} + (\hat{\alpha} + \hat{\beta}_{\text{Wh}}) e_{\text{Wh}}$$

So the coefficients have no intrinsic meaning. sklearn.LinearRegression() will pick some arbitrary coefficients, but it could just as well pick others.

If we want to interpret the coefficients, we should write the model using linearly independent features. We can discard whatever feature we like, as long as we get a linearly independent set. For example, let's use

$$y \approx \alpha\, 1 + \beta_{\text{As}} e_{\text{As}} + \beta_{\text{Bl}}\, e_{\text{Bl}} + \beta_{\text{Mi}}\, e_{\text{Mi}} + \beta_{\text{Oth}}\, e_{\text{Oth}}$$

This is similar to an example from earlier:

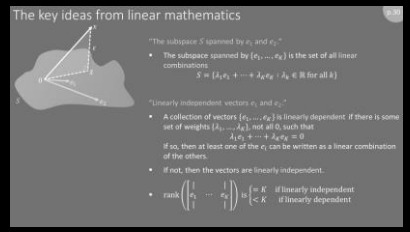
Exercise.
Are the following three vectors linearly independent? If not, find a subset that is.

$$e_1 = [1,1,1,1]$$
$$e_2 = [0,1,1,0]$$
$$e_3 = [1,0,0,1]$$

Here's an obvious linear relationship:

$1 = e_{\text{As}} + e_{\text{Bl}} + e_{\text{Mi}} + e_{\text{Oth}} + e_{\text{Wh}}$

From general linear maths: if the feature vectors are linearly independent, then there's a unique fit.



# Interpreting coefficients, by writing out predicted values

For the model

$$y \approx \alpha\,\mathbf{1} + \beta_{\mathrm{As}} e_{\mathrm{As}} + \beta_{\mathrm{Bl}}\, e_{\mathrm{Bl}} + \beta_{\mathrm{Mi}}\, e_{\mathrm{Mi}} + \beta_{\mathrm{Oth}}\, e_{\mathrm{Oth}}$$

what do the coefficients mean? A good way to see this is to write out predictions for generic datapoints:

| | | |
|---|---|---|
| For a person with | eth = "Asian", | predicted $y$ = $\alpha + \beta_{As}$ |
| | "Black" | $\alpha + \beta_{Bl}$ |
| | "mixed" | $\alpha + \beta_{Mi}$ |
| | "Other" | $\alpha + \beta_{oth}$ |
| | "White" | $\alpha$ |

Thus, $\alpha$ = predicted $y$ for a person with eth = "White"

$\beta_{Bl}$ = prediction (eth = "Black") — prediction (eth = "White").

Fitting the linear model $y \approx \alpha + \beta_{\texttt{eth}}$ using one-hot coding to extract the $\beta$ coefficients:

code: Azure notebook

```
ethnicity_levels = numpy.unique(df['eth'])
eth_onehot = [df['eth']==i for i in ethnicity_levels]

model = sklearn.linear_model.LinearRegression()
model.fit(numpy.column_stack(eth_onehot), df['y'])
α,βs = model.intercept_, model.coef_

print(f'α = {α}')
for i,β in zip(ethnicity_levels, βs):
    print(f'β[{i}] = {β}')
```

```
α = 171497104.17042407
β[Asian] = -171497103.3887985
β[Black] = -171497103.40795302
β[Mixed] = -171497103.2539072
β[Other] = -171497103.40197015
β[White] = -171497103.41569293
```

```
want_levels = ['Asian','Black','Mixed','Other']
eth_onehot = [df['eth']==i for i in want_levels]

model = sklearn.linear_model.LinearRegression()
model.fit(numpy.column_stack(eth_onehot), df['y'])
α,βs = model.intercept_, model.coef_

print(f'α = {α:.3}')
for i,β in zip(want_levels, βs):
    print(f'β[{i}] = {β:.3}')
```

```
α = 0.7547
β[Asian] = 0.02745
β[Black] = 0.00775
β[Mixed] = 0.1625
β[Other] = 0.0135
```

← predicted y (eth = "White")

← predicted y (eth = "Black") − predicted y (eth = "White").

Since $\beta_{\text{Black}} > 0$, predicted y (eth = "Black") > predicted y (eth = "White")

So there's no evidence that police are biased against eth = "Black".

The Guardian's analysis of the same dataset we looked at, is actually answering a different question to ours. Can you read a piece of journalism / polemic and decipher its model / assumptions?

Met police 'disproportionately' u

theguardian.com/law/2019/jan/26/met-police-disproportionately-use...



The Guardian



Stop and search

This article is more than 8 months old

# Met police 'disproportionately' use stop and search powers on black people

**London's minority black population targeted more than white population in 2018 – official figures**


**Vikram Dodd** *Police and crime correspondent*
Sat 26 Jan 2019 06.00 GMT

The Metropolitan police increased its use of stop and search last year, with a 19% rise among London's minority black population, which was targeted more than the white population, official figures show.

Analysis commissioned by the Guardian also shows that searches of black people were less likely to detect crime than those conducted on white people, and most stops found no wrongdoing.

Black people make up 15.6% of London's population while white people make up 59.8%. In 2018, 43% of searches were of black people, while 35.5% were of white people , according to official figures from the London Mayor's Office for Policing and Crime (MOPAC).

**Question 11.** For the police stop-and-search dataset in section 2.5 example 2.4, we wish to investigate intersectionality in police bias. We propose the linear model

$$1[\text{outcome} \neq \text{"False"}] \approx \alpha_{\text{gender}} + \beta_{\text{eth}}.$$

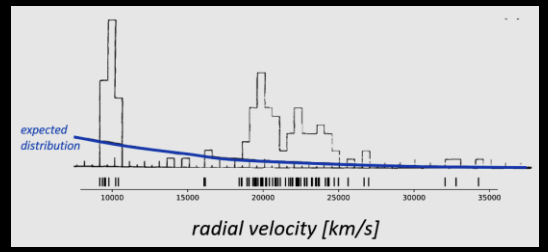
Write this as a linear model using one-hot coding. Are the parameters identifiable? If not, rewrite the model so they are, and interpret the parameters of your model.

*Hint. Recall our earlier exercise …*

Exercise.

Are the following five vectors linearly independent? If not, find a subset that is.

$$e_1 = [1,1,1,1]$$
$$e_2 = [0,1,1,0]$$
$$e_3 = [1,0,0,1]$$
$$e_4 = [1,1,1,0]$$
$$e_5 = [0,0,0,1]$$

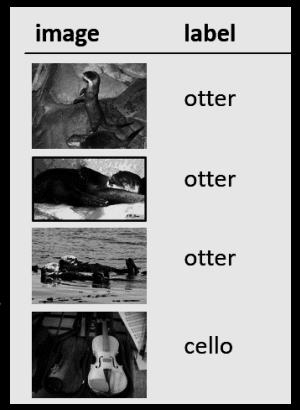



*clustering of galaxies (lecture 2) five parameters*

science = finding patterns in nature

- meaningful parameters
- interpretable patterns

*image classification (lecture 3) using a black-box function with billions of parameters*




modern machine learning = finding patterns in nature/data

- training a neural net = maximum likelihood estimation
- billions of parameters
- patterns beyond our comprehension

Scientists and social scientists have so far rejected neural networks. It's a crisis for science, that the models that have the potential to work best—these models are uninterpretable hence unreliable.