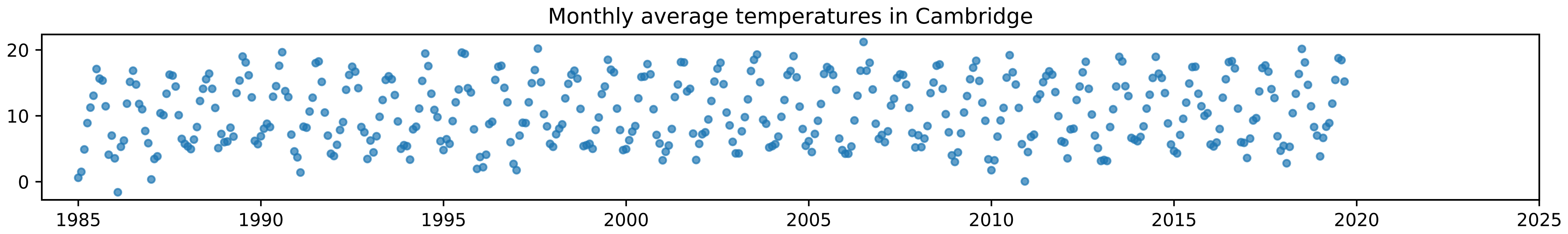
How can we estimate the rate of temperature increase from this dataset?



station	yyyy	mm	t	af	rain	sun	tmin	tmax	temp
Cambridge	1985	1	1985.00	23	37.3	40.7	-2.2	3.4	0.6
Cambridge	1985	2	1985.08	13	14.6	79	-1.9	4.9	1.5
Cambridge	1985	3	1985.16				1.1	8.7	4.9
⋮									

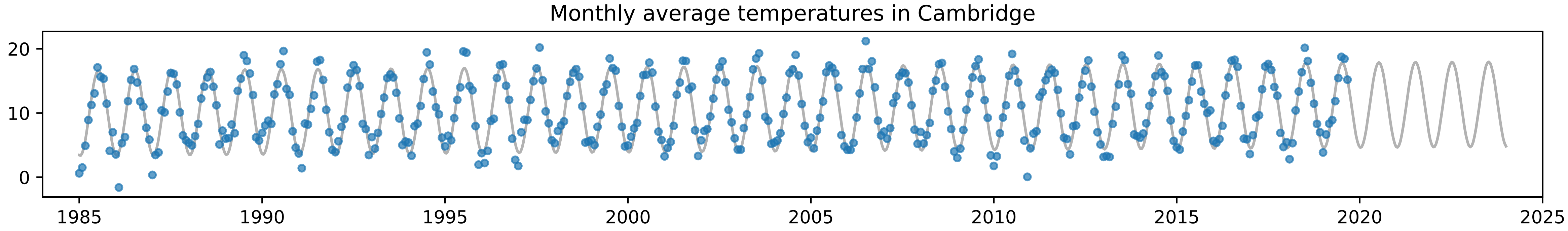
her station in Cambridge

What's a piece of code that might have generated these datapoints?

General method

1. Invent a probability model i.e. a simulator, parameterized by what we want to learn
2. Write out the likelihood
3. Maximize it

How can we estimate the rate of temperature increase from this dataset?



station	yyyy	mm	t	af	rain	sun	tmin	tmax	temp
Cambridge	1985	1	1985.00	23	37.3	40.7	-2.2	3.4	0.6
Cambridge	1985	2	1985.08	13	14.6	79	-1.9	4.9	1.5
Cambridge	1985	3	1985.16	[illegible]	[illegible]	[illegible]	1.1	8.7	4.9
⋮									

her station in Cambridge

In lectures, we proposed a sinusoid, scaled and shifted, plus randomness.

```
df = pandas.read_csv('https://teachingfiles.blob.core.windows.net/datasets/climate.csv')

k = 10
phase = 0.3
sigma = 5
offset = 10

temp = numpy.random.normal(loc = k * numpy.sin(2*π*df['t']+phase) + offset, scale=sigma)
```

1.6. Supervised learning

Consider a dataset in which each record stores several values. Often we think of each record i as having

- one value y_i as the *label* / *response variable*
- the others x_i (a tuple) as *predictors* / *covariates*

In *supervised learning* / *regression modelling*, we want to understand how the response depends on the covariates.

Method

1. Set up a probability model for a random variable Y_i, involving both x_i and unknown parameters
2. Treat y_i as a sample from Y_i, all records independent
3. Learn the unknown parameters using maximum likelihood estimation

Regression modelling / supervised	**Generative modelling / unsupervised**
Each record i consists of (x_i, y_i)	Each record i consists of x_i
Unknown parameters θ	Unknown parameters θ
Records are seen as independent samples	Records are seen as independent samples
Probability model $\mathrm{Pr}_Y(y_i \| x_i, \theta)$	Probability model $P_X(x_i \| \theta)$
Learn θ using mle	Learn θ using mle

Exercise 1.10 (Straight-line fit).
Given a labelled dataset $[(y_1, x_1), \dots, (y_n, x_n)]$ consisting of pairs of real numbers, consider the model

$$Y_i \sim \mathrm{Normal}(a + b\, x_i, \sigma^2)$$

where σ is given and a and b are unknown. Find maximum likelihood estimators for a and b. Also called "fitting the model"

From Lecture 2: if you take a Normal r.v. and scale/shift it, you get another Normal r.v.

$$Y_i = a + b x_i + N(0, \sigma^2)$$

$$\Pr_Y(y_i \mid x_i; a, b) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_i - (a + b x_i))^2 / 2\sigma^2}$$

$$\Pr_i(y \mid a, b)$$

alternative notation for the same thing..

- I like to write the density with a generic "y" since it describes the chance of any possible outcome
- I've supressed x_i and σ since they're known constants.

Keep y_i (the observed value of r.v.)
a, b (unknown parameters)

$$\mathrm{lik}(a, b \mid y_1, \cdots y_n) = \prod_{i=1}^{n} \Pr_i(y_i \mid a, b)$$

$$\mathrm{loglik}(a, b \mid y_1, \cdots, y_n) = \sum_{i=1}^{n} \log \Pr_i(y_i \mid a, b)$$

Pick a, b to maximize this.

```
import scipy.stats

σ = 1.5
def loglik(θ, x, y):
    a,b = θ
    lik = scipy.stats.norm.pdf(y, loc=a+b*x, scale=σ)
    return numpy.sum(numpy.log(lik))

initial_guess = [0,1]
(ahat,bhat) = scipy.optimize.fmin(lambda θ: -loglik(θ, x, y), initial_guess)
```

Exercise 1.11 (Binomial regression).
The UK Home Office makes available several datasets of police records, including stop-and-search data. It has been preprocessed to list the number of stops, and the number of those stops that led to the police finding something suspicious. Fit the model

$$Y_i \sim \mathrm{Bin}(x_i, p)$$

where $x_i = \mathrm{stops}$, $y_i = \mathrm{find}$, and p is the parameter to estimate.

police_force	year	stops	find
cleveland	2017	491	189
north-yorkshire	2017	742	237
leicestershire	2016	1475	518
…			

Exactly the same style of reasoning as for the preceeding exercise.

See printed lecture notes for the calculation.

Example 1.12 (Classification).

The ImageNet dataset has 14 million images, each hand-labelled with a category. I want to predict the label of an arbitrary image. I've build a black-box function that gives me scores for each possible category,

$$f: (x, \theta) \mapsto (s_1, s_2, \dots, s_K) \in \mathbb{R}^K$$

where

- x is an image (stored as an array of pixels)
- θ is a vector of parameters
- K is the number of possible categories
- $s_k = s_k(x, \theta)$ is the score for "image x having category k"
- x_i is image i in the dataset,
and $y_i \in \{1, \dots, K\}$ is its true category

How can I train this?

image	**label**
	otter
	otter
	otter
	cello
	otter
	cello

We want to see labels as sampled from a r.v. Y whose dist. depends on ~~x, θ~~ $f(x, \theta)$.

Try: $$Y = \begin{cases} 1 & \text{with prob. } s_1(x,\theta) \\ 2 & \text{with prob. } s_2(x,\theta) \\ \vdots & \end{cases}$$

The s_k are in $\mathbb{R}$. Not suitable as probabilities.

Try: $$Y = \begin{cases} 1 & \text{with prob. } \dfrac{e^{s_1(x,\theta)}}{e^{s_1(x,\theta)} + \dots + e^{s_k(x,\theta)}} \\ \vdots & \\ k & \text{with prob. } \dfrac{e^{s_k(x,\theta)}}{e^{s_1(x,\theta)} + \dots + e^{s_k(x,\theta)}} \end{cases}$$

$$\log \text{lik}(\theta \mid y_1, \dots, y_n) = \sum_{i=1}^{n} \log \left(\frac{e^{s_{y_i}(x_i,\theta)}}{e^{s_1(x_i,\theta)} + \dots + e^{s_k(x_i,\theta)}} \right)$$

1.7. Supervised learning and prediction loss *

Maximum likelihood estimation

$$\max_{\theta} \sum_{i=1}^{n} \log \frac{e^{s_{y_i}(x_i,\theta)}}{e^{s_1(x_i,\theta)} + \cdots + e^{s_K(x_i,\theta)}}$$

is equivalent to "loss minimization with the softmax cross-entropy loss function"

$$\min_{\theta} \sum_{i=1}^{n} \sum_{k=1}^{K} 1_{y_i=k} \log \left\{ \text{softmax}\big(s(x_i,\theta)\big)_k \right\}$$

ie it has the form

$$\min_{\theta} \sum_{i=1}^{n} \text{loss}\left(y_i,\ \text{output of black box } (x_i,\theta) \right)$$

Deep learning is all just maximum likelihood estimation.
Or, it's a toolbox of formulae and functions you have to learn.

A Brief Overview of Loss Functio

medium.com/udacity-pytorch-challengers/a-brief-overview-of-loss-functions-in-pytorch-c0ddb78...

Udacity PyTorch Challengers

Sign in Get started

What are loss functions?

Training the neural network is similar to how humans learn. We give data to the model, it predicts something and we tell it whether the prediction is correct or not. The model then corrects its mistakes. The model does this repeatedly until it reaches a certain level of accuracy, decided by us. Telling the model that the prediction was wrong is crucial for it to learn well. This where the loss function comes in. It tells the model how far off its estimation was from the actual value. While communicating with a human is easier, to tell so to a machine we need a medium(pun intended). This communication needs a ***how*** and a ***what***. The ***How*** is the programming language Python and the ***What*** is the Mathematics. In this post, I'll go through some *Hows*, *Whats* and the intuition behind them.

Smooth L1 Loss

`torch.nn.SmoothL1Loss`

Also known as Huber loss, it is given by —

$$\mathrm{loss}(x, y) = \begin{cases} 0.5(x - y)^2, & \text{if } |x - y| < 1 \\ |x - y| - 0.5, & \text{otherwise} \end{cases}$$

What does it mean?

It uses a squared term if the absolute error falls below 1 and an absolute

Example sheet 1
for supervisions

MRes AI4ER: you're not assessed, but you should do the work, email it to me, come to my office hours. Fridays, 2 – 4pm.

PhD: model solutions are available.

Example sheet 1
Learning with probability models
Foundations of Data Science—DJW—2019/2020

Question 1. Sketch the cumulative distribution function, and calculate the density function, for this continuous random variable:

```
def rx():
    u = random.random()
    return u * (1−u)
```

[Hint. See Exercise 3.3, from lecture 2.]

Question 2. We wish to implement a random variable whose cumulative distribution function $F(x) = \mathbb{P}(X \le x)$ is given by the function below. Here, a and b are parameters in the range $[0, 1]$. Sketch $F(x)$, and give code to generate such a random variable.

$$F(x) = \begin{cases} 0 & \text{if } x < 0 \\ bx/a & \text{if } 0 \le x \le a \\ b + (1-b)(x-a)/(1-a) & \text{if } a < x \le 1 \\ 1 & \text{if } x > 1. \end{cases}$$

[Hint. See slide 10 from lecture 2. All lecture slides are on Moodle.]

Question 3. Given a dataset $(x_1, \dots, x_n)$, we wish to fit a Poisson distribution. This is a discrete random variable with a single parameter $\lambda > 0$, called the rate, and

$$\Pr(x \mid \lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \quad \text{for } x \in \{0, 1, 2, \dots\}.$$

Show that the maximum likelihood estimator for λ is $\hat{\lambda} = n^{-1} \sum_{i=1}^n x_i$. *[Hint. This is a question about learning generative models. See section 1.5 exercise 1.7.]*

Question 4. Given a dataset [3,2,8,1,5,0,8], we wish to fit a Poisson distribution. Give code to achieve this fit, using scipy.optimize.fmin. *[Hint. See section 1.2 exercise 1.4.]*

Question 5. Given a dataset $(x_1, \dots, x_n)$, we wish to fit the Uniform$[0, \theta]$ distribution, where θ is unknown. By writing the density with explicit boundaries,

$$\Pr(x \mid \theta) = \frac{1}{\theta} 1_{x \ge 0} 1_{x \le \theta} \quad \text{for } x \in \mathbb{R},$$

show that the maximum likelihood estimator is $\hat{\theta} = \max_i x_i$.

Hint. In any question where the range of the random variable depends on unknown parameters, it's a good idea to include the boundaries explicitly in your density function, using an indicator function. See lecture 2 slides 10–11. A neat thing about indicator functions is that

$$1_{\xi \ge a} \times 1_{\xi \ge b} = 1_{\xi \ge a \text{ and } \xi \ge b} = 1_{\xi \ge \max(a,b)}.$$

Question 6 (A/B testing). Your company has two systems which it wishes to compare, A and B. It has asked you to compare the two, on the basis of performance measurements $(x_1, \dots, x_m)$ from system A and $(y_1, \dots, y_n)$ from system B. Any fool using Excel can just compare the averages, $\bar{x} = m^{-1} \sum_{i=1}^m x_i$ and $\bar{y} = n^{-1} \sum_{i=1}^n y_i$, but you are cleverer than that and you will harness the power of Machine Learning.

Supplementary questions
for revision or further study

Supplementary question sheet 1
Learning with probability models
Foundations of Data Science—DJW—2019/2020

*These questions are not intended for supervision (unless your supervisor directs you otherwise). Some of them are longer form exam-style questions, which you can use for revision. Others, labelled *, ask you to think outside the box.*

Question 12 (Cardinality estimation).

(a) Let T be the maximum of m independent Uniform$[0, 1]$ random variables. Show that $\mathbb{P}(T \le t) = t^m$. Find the density function $\Pr_T(t)$. *Hint. For two independent random variables U and V,*

$$\mathbb{P}(\max(U, V) \le x) = \mathbb{P}(U \le x \text{ and } V \le x) = \mathbb{P}(U \le x)\,\mathbb{P}(V \le x).$$

(b) A common task in data processing is counting the number of unique items in a collection. When the collection is too large to hold in memory, we may wish to use fast approximation methods, such as the following: Given a collection of items $a_1, a_2, \dots$, compute the hash of each item $x_1 = h(a_1), x_2 = h(a_2), \dots$, then compute $t = \max_i x_i$.

If the hash function is well designed, then each x_i can be treated as if it were sampled from Uniform$[0, 1]$, and unequal items will yield independent samples..

The more unique items there are, the larger we expect t to be. Given an observed value t, find the maximum likelihood estimator for the number of unique items. *[Hint. This is about finding the mle from a single observation, as in exercise 1.1.]*

http://blog.notdot.net/2012/09/Dam-Cool-Algorithms-Cardinality-Estimation

Question 13*. Sketch the cumulative distribution functions for these two random variables. Are they discrete or continuous?

```
def rx():
    u = random.random()
    return 1/u
def ry():
    u2 = random.random()
    return rx() + math.floor(u2)
```

[Hint. For intuition, use simulation. Generate say 10,000 samples, and plot a histogram, then a plot of "how many are ≤ x" as a function of x.]

Question 14. A point lightsource at coordinates $(0, 1)$ sends out a ray of light at an angle Θ chosen uniformly in $[-\pi/2, \pi/2]$. Let X be the point where the ray intersects the horizontal line through the origin. What is the density of X? *[Hint. See exercise 3.3, from lecture 2.]*

Note: This random variable is known as the Cauchy distribution. It is unusual in that it has no mean.

Mock exam question 1

Friday's lecture will be an OPTIONAL examples class, working through this question.

COMPUTER SCIENCE TRIPOS Part IB – mock – Paper 6

1 Foundations of Data Science (DJW)

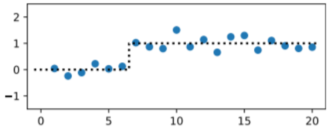
(a) A 0/1 signal is being transmitted over a noisy channel. The transmitted signal at timeslot $i \in \{1, \dots, n\}$ is $x_i \in \{0, 1\}$, and furthermore we know that this signal starts at 0 and then flips to 1, i.e. there is a parameter $\theta \in \{1, \dots, n-1\}$ such that

$$x_i = \begin{cases} 0 & \text{for } i \le \theta, \\ 1 & \text{for } i > \theta, \end{cases}$$

but the value of θ is unknown. The channel is noisy, and the received signal in timeslot i is

$$Y_i \sim x_i + \text{Normal}(0, \varepsilon^2)$$

where ε is known.



(i) Given received signals $(y_1, \dots, y_n)$, find an expression for the log likelihood, $\log \text{lik}(\theta \mid y_1, \dots, y_n)$. [5 marks]

(ii) Give pseudocode for finding the maximum likelihood estimator $\hat{\theta}$. [3 marks]

(b) The Gaussian Mixture Model with m components can be written as a two-stage random variable: first generate $K \in \{1, \dots, m\}$, $\mathbb{P}(K = k) = p_k$, then generate $X \sim \text{Normal}(\mu_K, \sigma_K^2)$. Here $p_1, \dots, p_m$ and $\mu_1, \dots, \mu_m$ and $\sigma_1, \dots, \sigma_m$ are unknown parameters, with $p_k > 0$ and $\sigma_k > 0$ for all k, and $p_1 + \dots + p_m = 1$.

2.1. Fitting a linear model

It's too much work to invent a new probability model from scratch every time. Instead, a good choice is linear models—a flexible and interpretable class of supervised learning models.

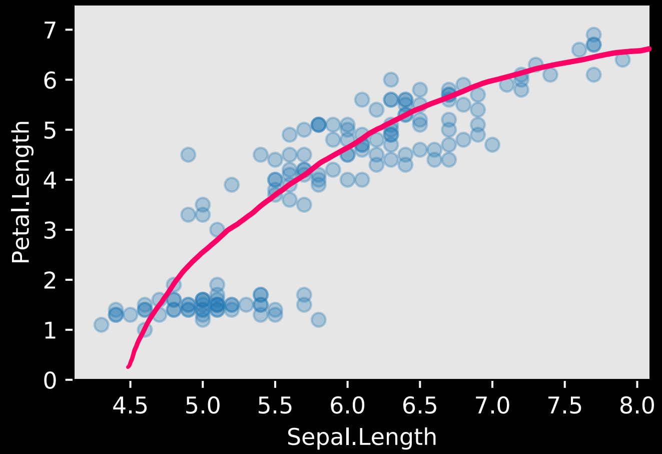
- predictor = vector of numerical *features*
- label = numerical response variable
- label depends on a *linear combination of features*

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 α, β, γ (to be estimated),

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

Features:

1, Sepal.length, Sepal.length^2

Label:

Petal.Length

Linear combination of features:

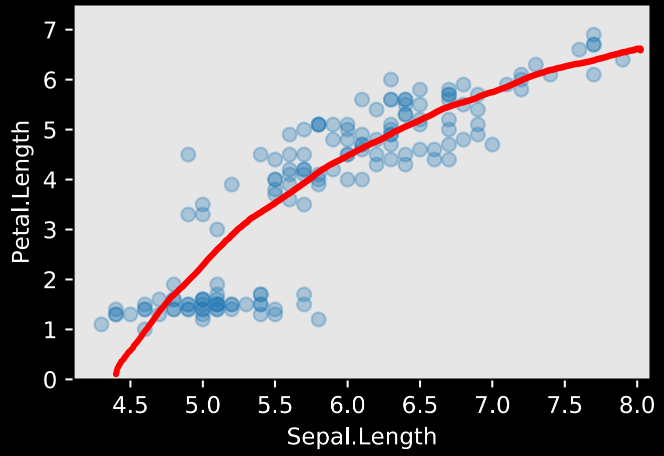
$\alpha\, 1 + \beta\, \text{Sepal.length} + \gamma\, \text{Sepal.length}^2$

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 α, β, γ (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

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

The feature vectors are

$\texttt{one} = [1, 1, \dots, 1]$

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

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

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

$$\begin{bmatrix} PL_1 \\ \vdots \\ PL_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} SL_1 \\ \vdots \\ SL_n \end{bmatrix} + \gamma \begin{bmatrix} SL_1^2 \\ \vdots \\ SL_n^2 \end{bmatrix}$$