Lecture 3 – Predictive Modeling (Regression)

Terminology, Linear and Polynomial Regression

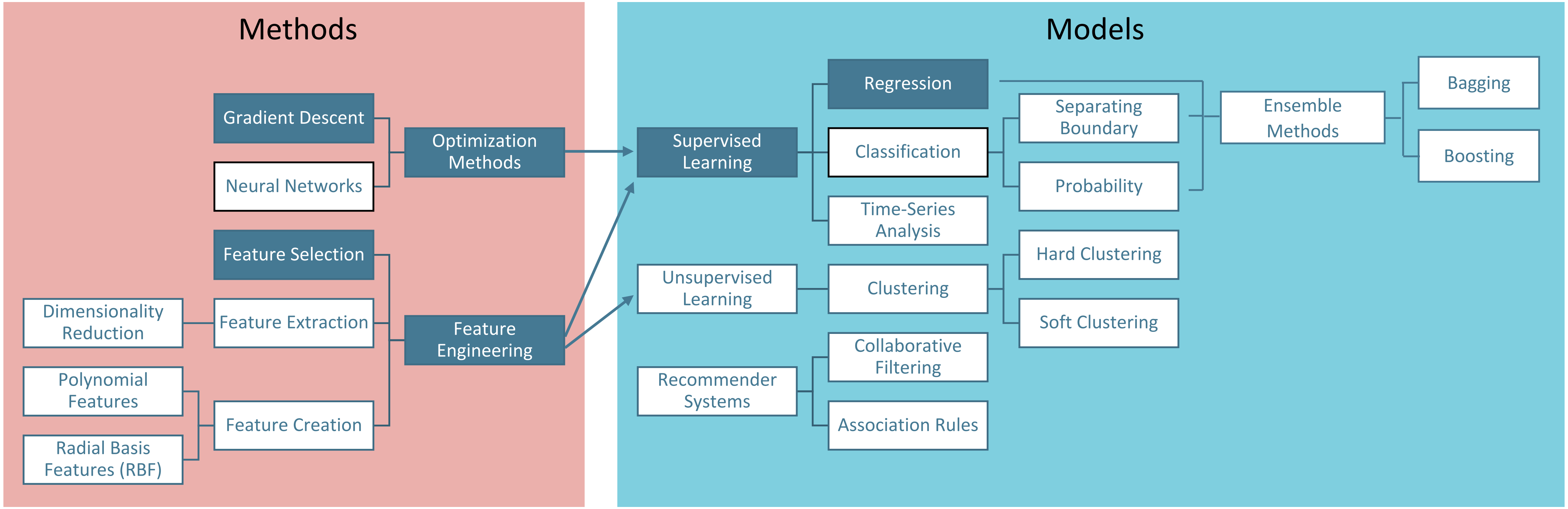
Information Systems for Sustainable Society (is3) | WiSo Faculty | Univ.-Prof. Dr. Wolfgang Ketter | October 25th 2023

Agenda

Linear Regression

Beyond Linearity – Polynomial Regression

Topic Structure – Today's Lecture

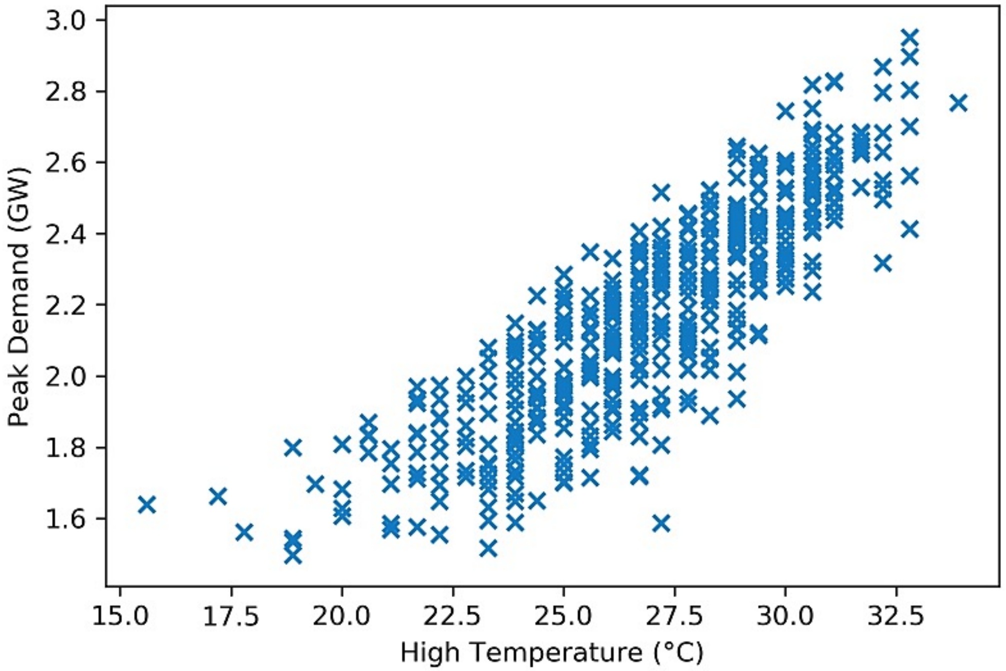


A simple example: predicting electricity use

Date	Average demand	Peak demand	High temperature	Average temperature
01.01.2013	1.598524	1.859947	0	-1.68
02.01.2013	1.809347	2.054215	-3.9	-6.58
03.01.2013	1.832822	2.04955	0.6	-6.12
04.01.2013	1.812699	2.008168	0	-1.95

- What will peak power consumption be in Pittsburgh tomorrow?
- Difficult to build an “a priori” model from first principles to answer this question
- But, relatively easy to record past days of consumption, plus additional features that affect consumption (i.e., weather)

Plot of consumption vs. temperature



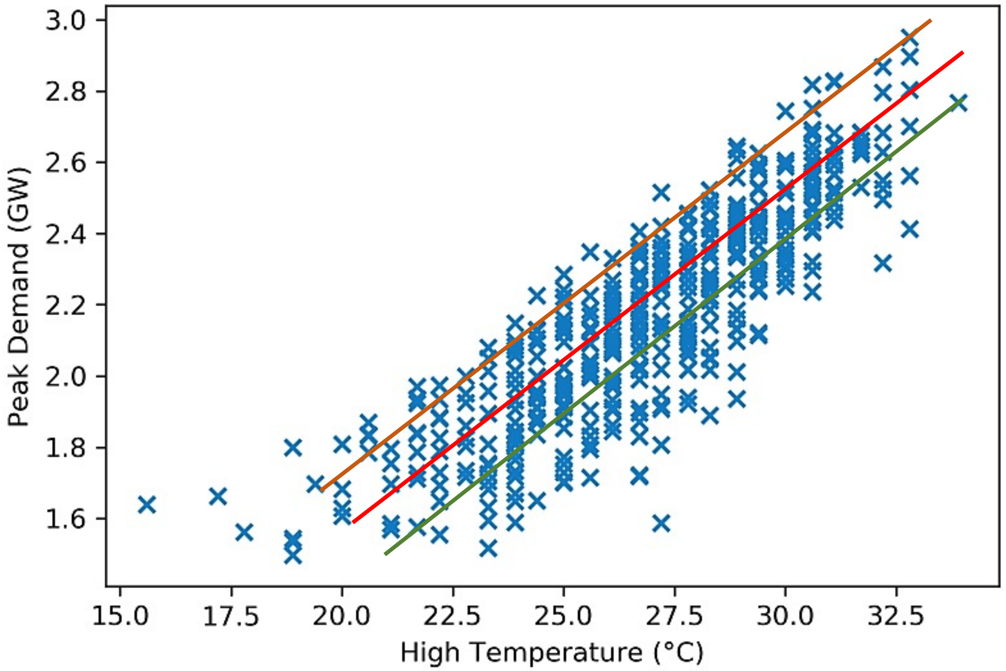
- Plot of high temperature vs. peak demand for summer months (June – August) for past six years

Hypothesis: linear model

$$Peak\_Demand \approx \theta_1 * High\_Temperature + \theta_2$$

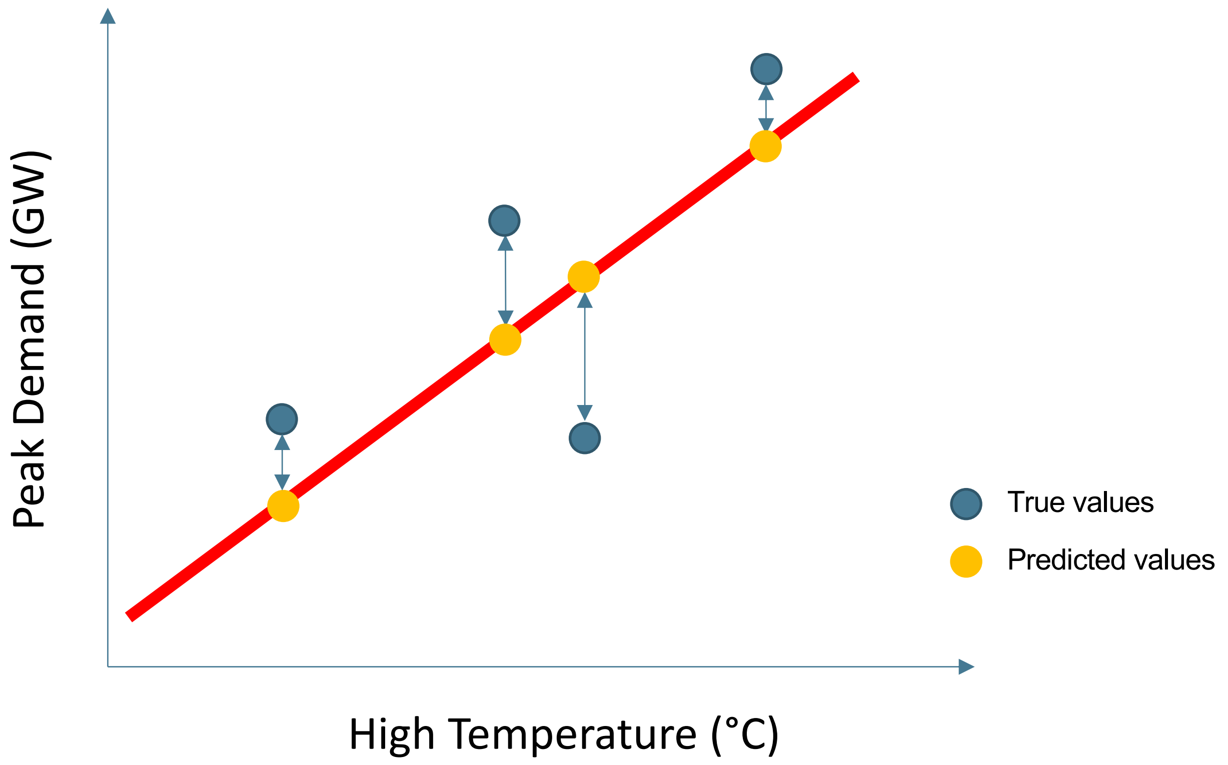
- Let's suppose that the peak demand approximately fits a linear model
  - Peak_Demand ≈ θ_1*High_Temperature + θ_2
- Here θ_1 is the "slope" of the line, and θ_2 is the intercept
- How do we find a "good" fit to the data?
  - Many possibilities, but natural objective is to minimize some difference between this line and the observed data

Plot of consumption vs. temperature



- Plot of high temperature vs. peak demand for summer months (June – August) for past six years

What is our goal? Why don't we just pick random parameters?



Intuition: Find parameters θ such that the distance between the resulting function h_θ and the data points are minimized.

How do we find parameters?

- How do we find the parameters θ_1 , θ_2 that minimize the objective function $E(\theta)$

$$E(\theta) = \sum_{i \in days} (\underbrace{\hat{y}^{(i)}}_{\text{Predicted value at } x^{(i)}} - \underbrace{y^{(i)}}_{\text{True value at } x^{(i)}})^2$$

$$= \sum_{i \in days} (\theta_1 High\_Temperature^{(i)} + \theta_2 - Peak\_Demand^{(i)})^2$$

$$= \sum_{i \in days} (\theta_1 x^{(i)} + \theta_2 - y^{(i)})^2$$

Let's recap the process of Machine learning tasks…

The basic process (supervised learning):

Training Data	ML algorithm	Predictions
$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$ $(x^{(3)}, y^{(3)})$ $\vdots$	Hypothesis function $y^{(i)} \approx h(x^{(i)})$	New example x $\hat{y} = h(x)$

- This has been an example of a machine learning algorithm
  - Basic idea: in many domains, it is difficult to hand-build a predictive model, but easy to collect lots of data; machine learning provides a way to automatically infer the predictive model from data

At the core of each ML algorithm lies the canonical machine learning optimization problem

The canonical machine learning optimization problem:

$$\underset{\theta}{Minimize} \sum_{i=1}^{m} \ell(h_\theta(x^{(i)}), y^{(i)})$$

- Virtually **every machine learning algorithm** has **this form**, just specify
  1. What is the **hypothesis** function?
  2. What is the **loss/cost (objective)** function?
  3. **How** do we **solve** the optimization problem?

Let us re-visit this simple machine learning task using the canonical formulation and matrix notation

- Using our new terminology of **hypothesis, objective and optimization**, plus matrix notion, let's revisit how to solve linear regression **with a squared error loss**
- **Setup:**
  - Linear **hypothesis function**: $h_\theta(x) = \sum_{j=1}^{n} \theta_j . x_j$
  - **Objective function** (squared loss): $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$
  - Resulting machine learning **optimization problem**:

$$\underset{\theta}{Minimize} \sum_{i=1}^{m} (\sum_{j=1}^{n} \theta_j . x_j^{(i)} - y^{(i)})^2 \equiv \underset{\theta}{Minimize} E(\theta)$$

where n = number of features; m=number of instances

(Multi-) Linear Regression

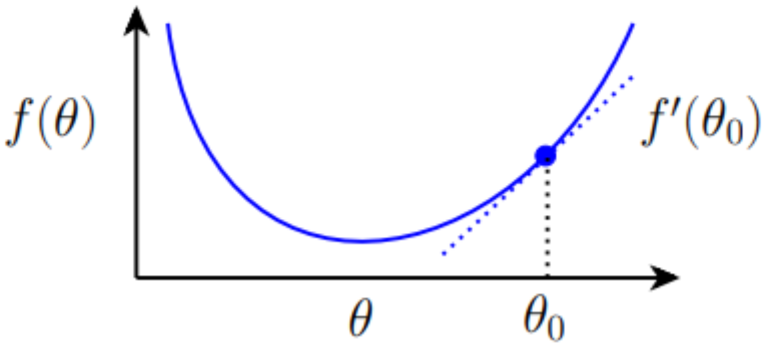
	The general case of MLR	Example for n = 2
Input features	$x^{(i)} \in \mathbb{R}^N, i = 1, \dots, m$	$x^{(i)} = \begin{bmatrix} High\_Temperature^{(i)} \\ 1 \end{bmatrix} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$
Target features	$y^{(i)} \in \mathcal{Y} \subseteq \mathbb{R}, i = 1, \dots, m$	$y^{(i)} \in \mathbb{R} = Peak\_Demand^{(i)}$
Model parameters	$\theta \in \mathbb{R}^N, \theta = (\theta_1, \dots, \theta_N)$	$\theta = (\theta_1, \theta_2)$
Hypothesis function	$h_\theta: \mathbb{R}^N_{\mid x_N^{(i)}=1} \to y,\ h_\theta(x) = \sum_{j=1}^{n} \theta_j . x_j$	$\hat{y}^{(i)} := h_\theta\left(x_1^{(i)}, x_2^{(i)}\right) = \theta_1 x_1^{(i)} + \theta_2$
Objective function	$\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$	$\ell(\hat{y}, y) = (\hat{y} - y)^2$

How do we find parameters?

- General idea: suppose we want to minimize some loss function $l(\theta)$

$$l(\theta) = \sum_{i \in days} (\theta_1 x^{(i)} + \theta_2 - y^{(i)})^2$$

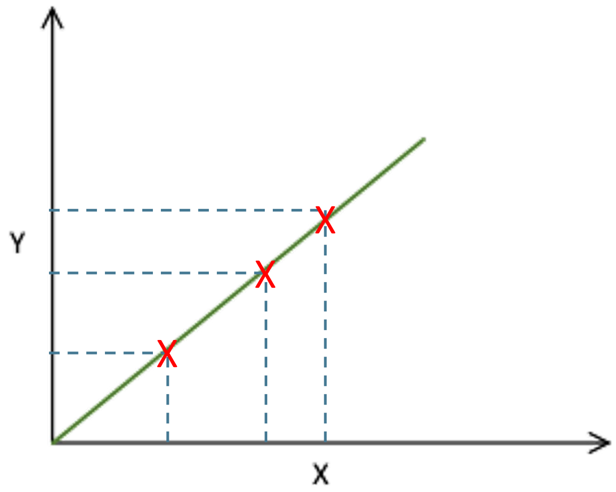
- Derivative is slope of the function, so **negative derivative** points "downhill"



Let's simplify the hypothesis function

To simplify the function, let's only consider one parameter: $h_\theta(x^{(i)}) = \theta_1 x^{(i)}$

For fixed θ_1, $h_\theta(x)$ is a function of x

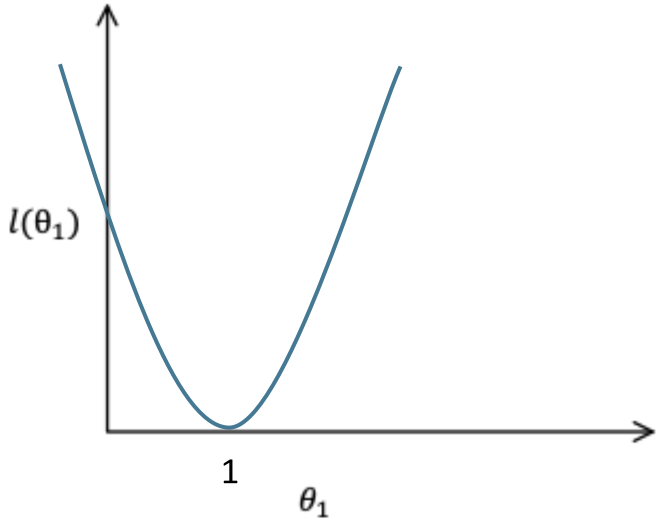


θ_1=1 → $h_\theta\left(x^{(i)}\right) = y^{(i)}$

$l(\theta_1{=}1) = \sum_{i=1}^{m}(h_\theta\left(x^{(i)}\right) - y^{(i)})^2 =$

$(0 + 0 + .. + 0)^2 = 0$

$l(\theta_1)$ is a function of θ_1



If we compute and visualize $l(\theta_1)$ for different values of θ_1, we observe that $l(\theta_1)$ is quadratic function

We can compute the **partial derivative** with respect to **an arbitrary model parameter θ_j** – Again we use simple rules of differentiation

$$\frac{\partial E(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} (\sum_{k=1}^{n} \theta_k x_k^{(i)} - y^{(i)})^2 \qquad (1)$$

$$= \sum_{i=1}^{m} \frac{\partial}{\partial \theta_j} (\sum_{k=1}^{n} \theta_k x_k^{(i)} - y^{(i)})^2 \qquad (2)$$

$$= \sum_{i=1}^{m} 2(\sum_{k=1}^{n} \theta_k x_k^{(i)} - y^{(i)}) \frac{\partial}{\partial \theta_j} \sum_{k=1}^{n} \theta_k x_k^{(i)} \qquad (3)$$

$$= 2 \sum_{i=1}^{m} \sum_{k=1}^{n} (\theta_k x_k^{(i)} - y^{(i)})\ x_j^{(i)} \qquad (4)$$

- We use the k subscript here as indicator for the features to avoid collision with the j indicator
- To get from (2) to (3) we apply the **chain rule**
  - h(x) = f(g(x))
  - h'(x) = f'(g(x)) g'(x)
- To get from (3) to (4) we realize that $\frac{\partial}{\partial \theta_j} \sum_{j=1}^{n} \theta_k x_k^{(i)}$ is only non-constant for k=j in which case it is $x_j^{(i)}$

Universität zu Köln

Finding the best θ – The gradient descent algorithm

To find a good value of θ, we can **repeatedly take steps in the direction of the negative partial derivatives** for each value of the error function.

1. **Initialize** $\theta_j := 0, \quad j = 1, \dots, n$
2. **Repeat:** For $j = 1, \dots, n$:

$$\theta_j := \theta_j - \underbrace{\boxed{\propto}}_{\text{learning rate}} \underbrace{\boxed{\sum_{i=1}^{m} \left(\sum_{k=1}^{n} \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)}}}_{\text{derivative}}$$

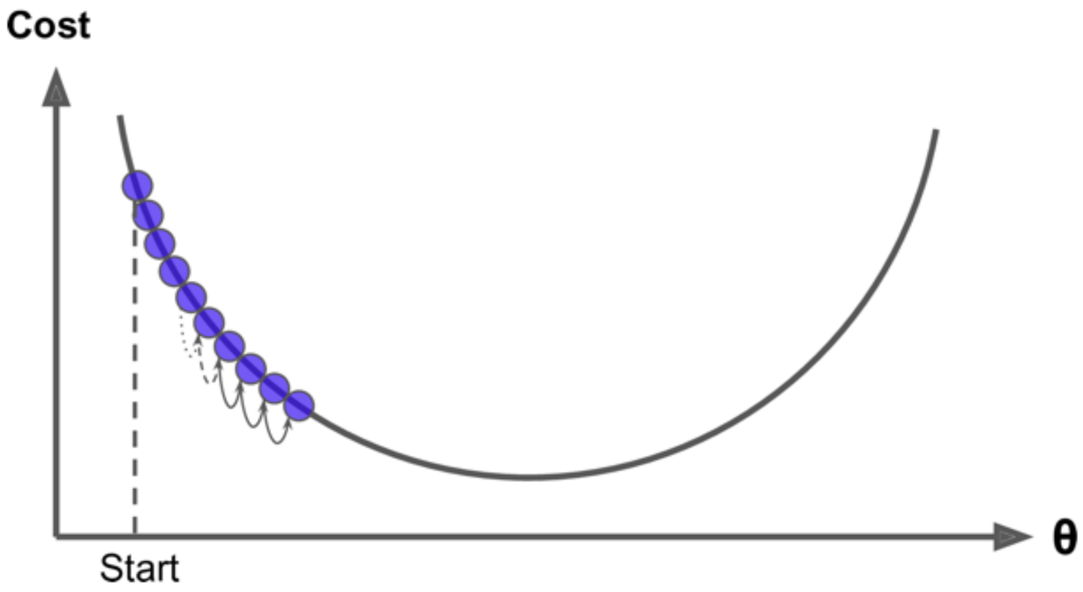
- We can **drop the constant factor 2** as it will be captured by the constant hyperparameters ∝ (the step size)
- Note: **do not actually implement it like this**, you'll **want to use the matrix/vector notation**

Some truths about gradient descent

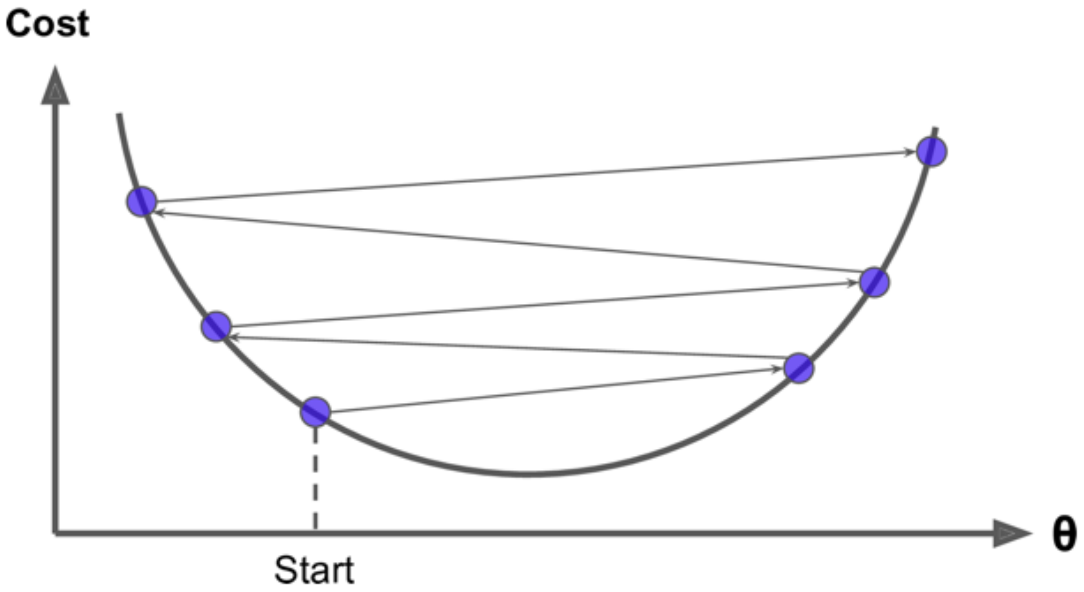
- Gradient descent is a greedy, heuristic algorithm
- The algorithm is **appealing in its generality** but certainly also has **some limitations**
  - Step size (learning rate) selection and tuning required
  - Potentially large number of iterations
  - Proper data normalization needed
  - No guarantee that global optimum is found
- These issues will be unavoidable for many of the problems we encounter
- But it turns out that **for least squares** in particular, **there is an alternative** that is much easier to compute in many cases

Gradient descent intuition

- If ∝ is too small, gradient descent can be slow.

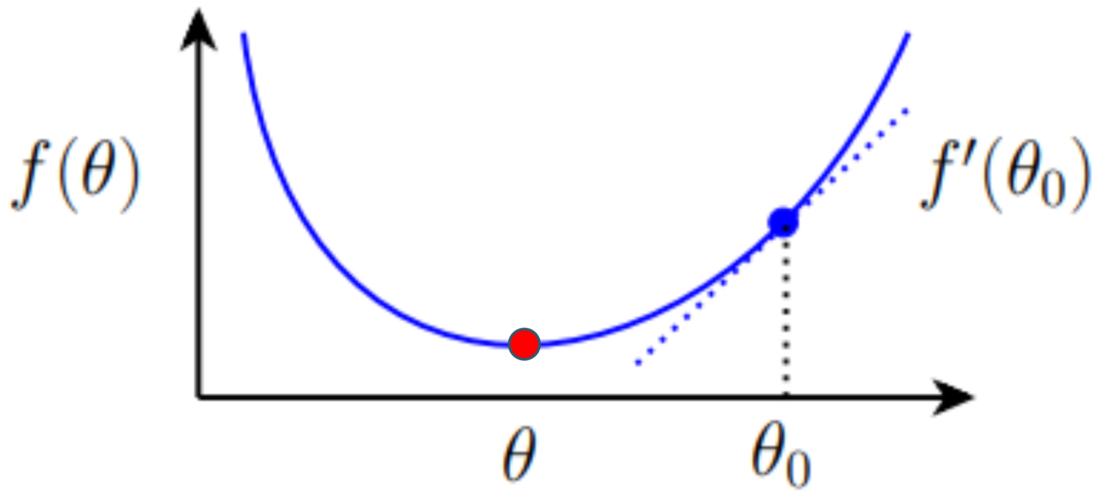


- If ∝ is too large, gradient descent can overshoot the minimum and fail to converge.



Solving least squares analytically – All we need to do is to set the gradient to 0

Visualizing the optimal solution



- Gradient also gives a condition for optimality
- The gradient of the error function must equal zero at which point minimum of the error function is reached (i.e. the optimal solution to the optimization function is found)

To do so let us first introduce a new term called the **gradient of a function**, a vector of all partial derivatives

- It is typically more convenient to work with a vector of all partial derivatives, called the **gradient**
- This way we can **drop all the summation signs** and **use linear algebra** to **find solutions efficiently**
- For a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, the gradient is a vector

$$\nabla_\theta f(\theta) = \begin{bmatrix} \frac{\partial f(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial f(\theta)}{\partial \theta_n} \end{bmatrix} \in \mathbb{R}^n$$

Defining the gradient makes it clearer how to analytically find the optimal solution for least squares

- We can actually **simplify** the gradient computation (both in notation and computationally) substantially using matrix/vector notation
- Recall that we have just derived the following solution for the partial derivative with respect to any parameter θ_j

$$\frac{\partial E(\theta)}{\partial \theta_j} = \sum_{i=1}^{m} \left(\sum_{k=1}^{n} \theta_k x_k^{(i)} - y^{(i)} \right) \, x_j^{(i)}$$

Since the only term that depends on j is $x_j^{(i)}$ we can write the full gradient of $E(\theta)$ as

$$\Leftrightarrow \; \nabla_\theta E(\theta) = \sum_{i=1}^{m} x^{(i)} \left(x^{(i)^T} \theta \; - \; y^{(i)} \right)$$

Solving for $\nabla_\theta E(\theta) = 0$ by carrying out some simple algebraic manipulation provides an analytical solution for the parameter vector θ

$$\nabla_\theta E(\theta) = 0 \quad \textbf{(1)}$$

$$\sum_{i=1}^{m} x^{(i)} \left(x^{(i)^T} \theta \; - \; y^{(i)} \right) = 0 \quad \textbf{(2)}$$

$$\Rightarrow \left(\sum_{i=1}^{m} x^{(i)} x^{(i)^T} \right) \theta - \sum_{i=1}^{m} x^{(i)} y^{(i)} = 0 \quad \textbf{(3)}$$

$$\Rightarrow \theta^* = \left(\sum_{i=1}^{m} x^{(i)} x^{(i)^T} \right)^{-1} \left(\sum_{i=1}^{m} x^{(i)} y^{(i)} \right) \quad \textbf{(4)}$$

- Solving for $\nabla_\theta E(\theta) = 0$ and carrying out some simple transformation provides an analytical solution for the parameter vector θ
- To get form (2) to (3) simply **multiply out the terms**
- To get form (3) to (4) **solve for $\boldsymbol{\theta}^*$** and re-arrange

We can simplify even further by using matrix notation to group the individual instances m – The result is known as the **normal equations**

- Let's define the matrices

$$X = \begin{bmatrix} - \ x^{(1)^T} \ - \\ - \ x^{(2)^T} \ - \\ \vdots \\ - \ x^{(m)^T} \ - \end{bmatrix} \qquad , y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

- Then:

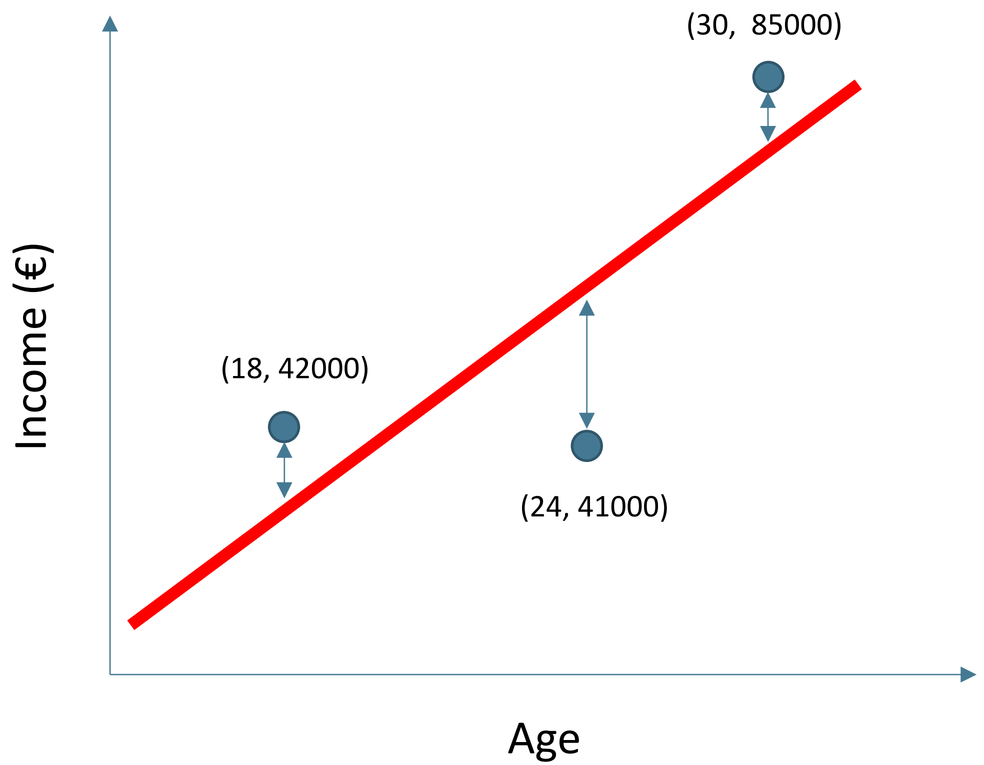
$$\nabla_\theta E(\theta) = \sum_{i=1}^{m} x^{(i)} \left(x^{(i)^T} \theta - y^{(i)} \right) = X^T (X\theta - y)$$

- Let $\nabla_\theta E(\theta) = 0$ for optimality:

$$\Rightarrow \theta^* = (X^T X)^{-1} X^T \mathrm{y}$$

- These are known as the **normal equations** an extremely convenient closed-form solution for least squares (without need for normalization)

A Simplified Linear Regression Example



- Assume linear model h(x) = $\theta_1 \, x + \theta_2$
- Find θ_1, θ_2 such that the squared error loss is minimized (based on the data below)
- Input:

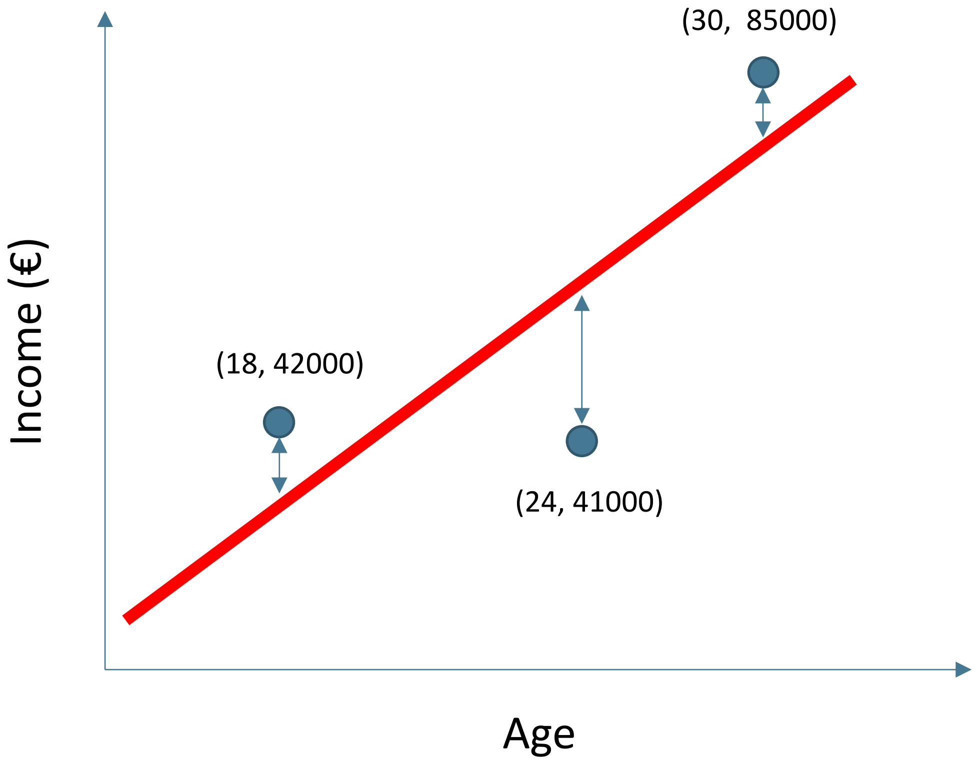
$$X = \begin{pmatrix} 18 & 1 \\ 24 & 1 \\ 30 & 1 \end{pmatrix} \qquad Y = \begin{pmatrix} 42000 \\ 41000 \\ 85000 \end{pmatrix}$$

- Intermediate steps:

$$X^T X = \begin{pmatrix} 18 & 24 & 30 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 18 & 1 \\ 24 & 1 \\ 30 & 1 \end{pmatrix} = \begin{pmatrix} 1800 & 72 \\ 72 & 3 \end{pmatrix}$$

$$(X^T X)^{-1} = \begin{pmatrix} 1/72 & -1/3 \\ -1/3 & 25/3 \end{pmatrix}$$

A Simplified Linear Regression Example



- Result:

$$\theta^* = (X^T X)^{-1} X^T Y =$$

$$= \begin{pmatrix} 1/72 & -1/3 \\ -1/3 & 25/3 \end{pmatrix} \begin{pmatrix} 18 & 24 & 30 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 42000 \\ 41000 \\ 85000 \end{pmatrix}$$

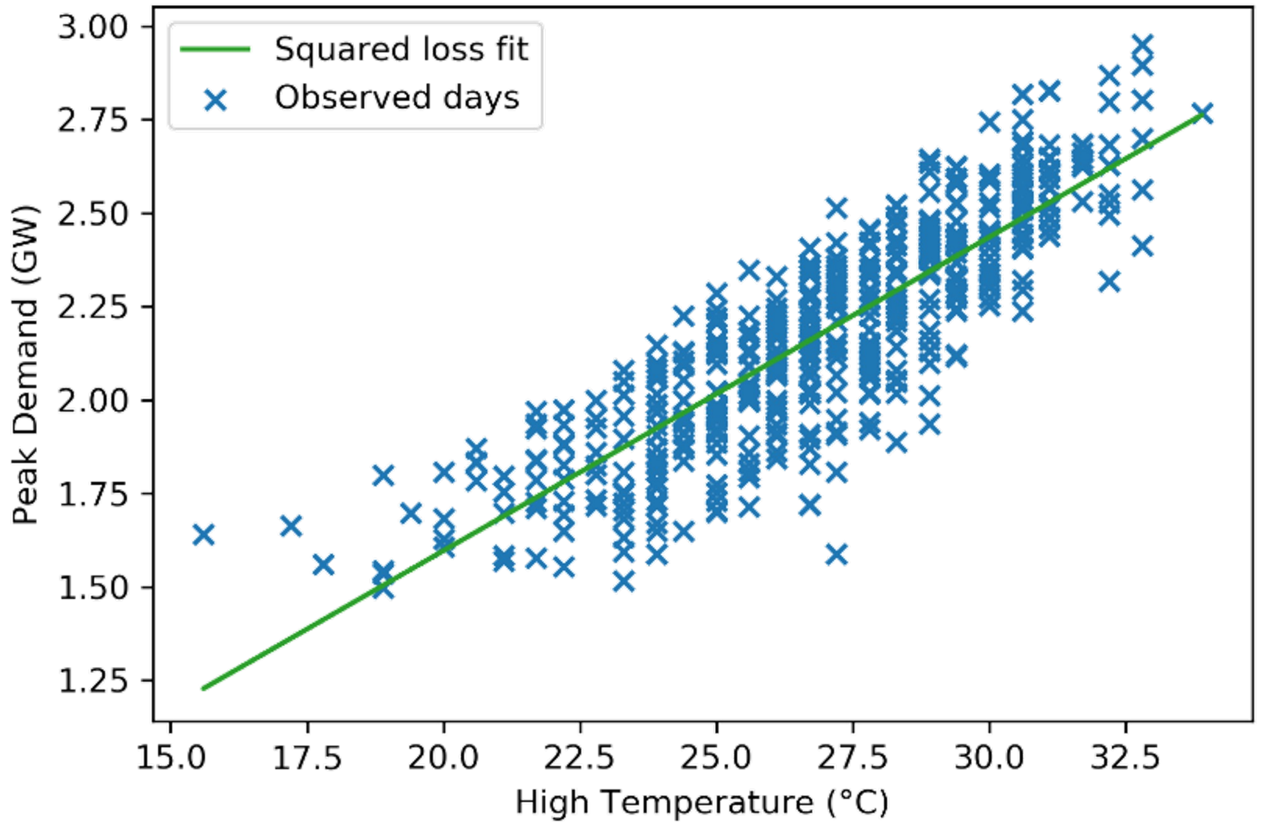
$$= \begin{pmatrix} -1/12 & 0 & 1/12 \\ 7/3 & 1/3 & -5/3 \end{pmatrix} \begin{pmatrix} 42000 \\ 41000 \\ 85000 \end{pmatrix}$$

$$= \begin{pmatrix} 10750/3 \\ -30000 \end{pmatrix}$$

- “Optimal“ Function:
  - h(x) = 10750/3 * x – 30000 = 10750/3 * “age“ – 30000

Returning to our electricity demand example: One predictive feature

- Using the closed form solution, we obtain

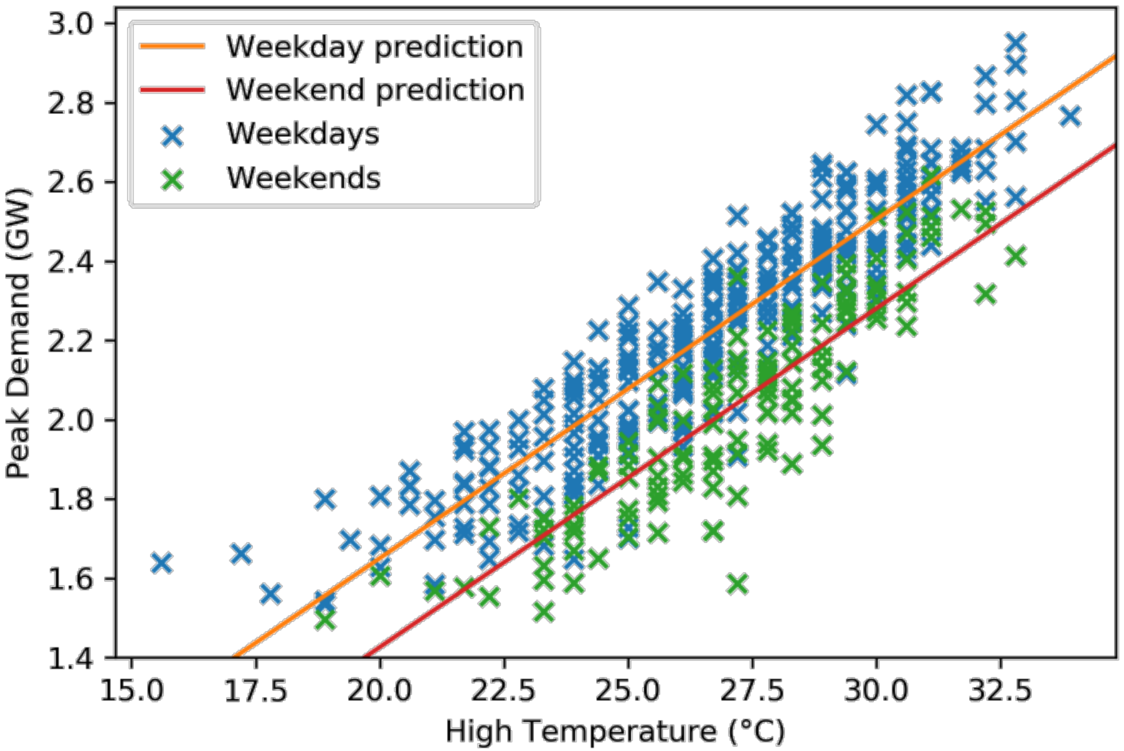


$$x^{(i)} = \begin{bmatrix} High\_Temperature^{(i)} \\ 1 \end{bmatrix}$$

$$\theta^* = (X^T X)^{-1} X^T Y = \begin{bmatrix} 0.084 \\ -0.080 \end{bmatrix}$$

$$\hat{y} = 0.084 * High\_Temp^{(i)} - 0.08$$

Returning to our electricity demand example: Two predictive features



- Feature vector

$$x^{(i)} = \begin{bmatrix} High\_Temperature^{(i)} \\ Is\_Weekday^{(i)} \\ 1 \end{bmatrix}$$

- Parameters derived by least squares

$$\Rightarrow \theta^{*} = (X^T X)^{-1} X^T y = \begin{bmatrix} 0.085 \\ 0.224 \\ -0.282 \end{bmatrix}$$

Five key components that make up a ML model

	Term	Example
Input features	$x^{(i)} \in \mathbb{R}^n, i = 1, \dots, m$	$x^{(i)} = \begin{bmatrix} High\_Temperature^{(i)} \\ Is\_Weekday^{(i)} \\ 1 \end{bmatrix}$
Target features	$y^{(i)} \in \mathcal{Y}, i = 1, \dots, m$	$y^{(i)} \in \mathbb{R} = Peak\_Demand^{(i)}$
Model parameters	$\theta \in \mathbb{R}^n$	$\theta = (\theta_1, \theta_2, \theta_3)$
Hypothesis function	$h_\theta: \mathbb{R}^n \rightarrow y$, predicts output given input	$h_\theta(x) = \sum_{j=1}^{n} \theta_j . x_j$
Objective function	$\ell: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, measures the difference between a prediction and an actual output	$\ell(\hat{y}, y) = (\hat{y} - y)^2$

In the previous example, we had the same slope for both weekend and weekday examples, just with a different intercept. Is it possible to have a model with both different slopes and different intercepts?

a) The previous example already **did** have different slopes
b) This is not possible with linear regression
c) You need to build two models, one just on weekdays and one just on weekends
d) You can do it with a single model, just with different features

In the previous example, we had the same slope for both weekend and weekday examples, just with a different intercept. Is it possible to have a model with both different slopes and different intercepts?

a) The previous example already **did** have different slopes
b) This is not possible with linear regression
c) You need to build two models, one just one weekdays and one just on weekends
d) You can do it with a single model, just with different features

Obviously it is possible to build two models for weekdays and weekends data and have different parameters. But, it is also possible to have one model for both data. In this case you need to adapt your feature vector X.

$$x^{(i)} = \begin{bmatrix} High\_Temperature^{(i)} Is\_Weekday^{(i)} \\ High\_Temperature^{(i)} (1 - Is\_Weekday^{(i)}) \\ Is\_Weekday^{(i)} \\ 1 \end{bmatrix}$$

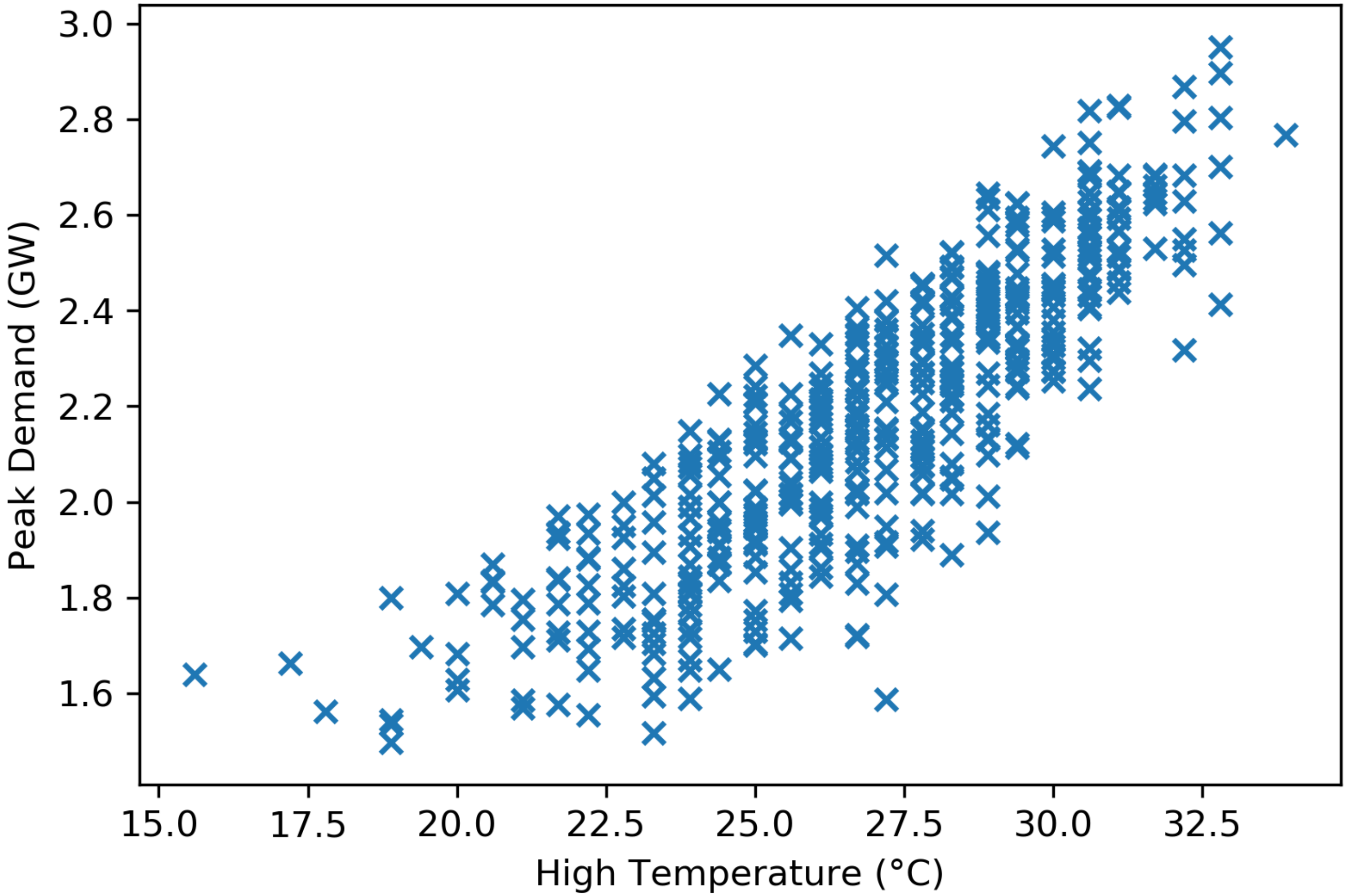
By forming the above X we would have different slopes for weekdays and weekends with training one model. Where *Is_Weekday* is a binary variable taking a value of 1 if the day is a weekday and a value of 0 if it is not.

Agenda

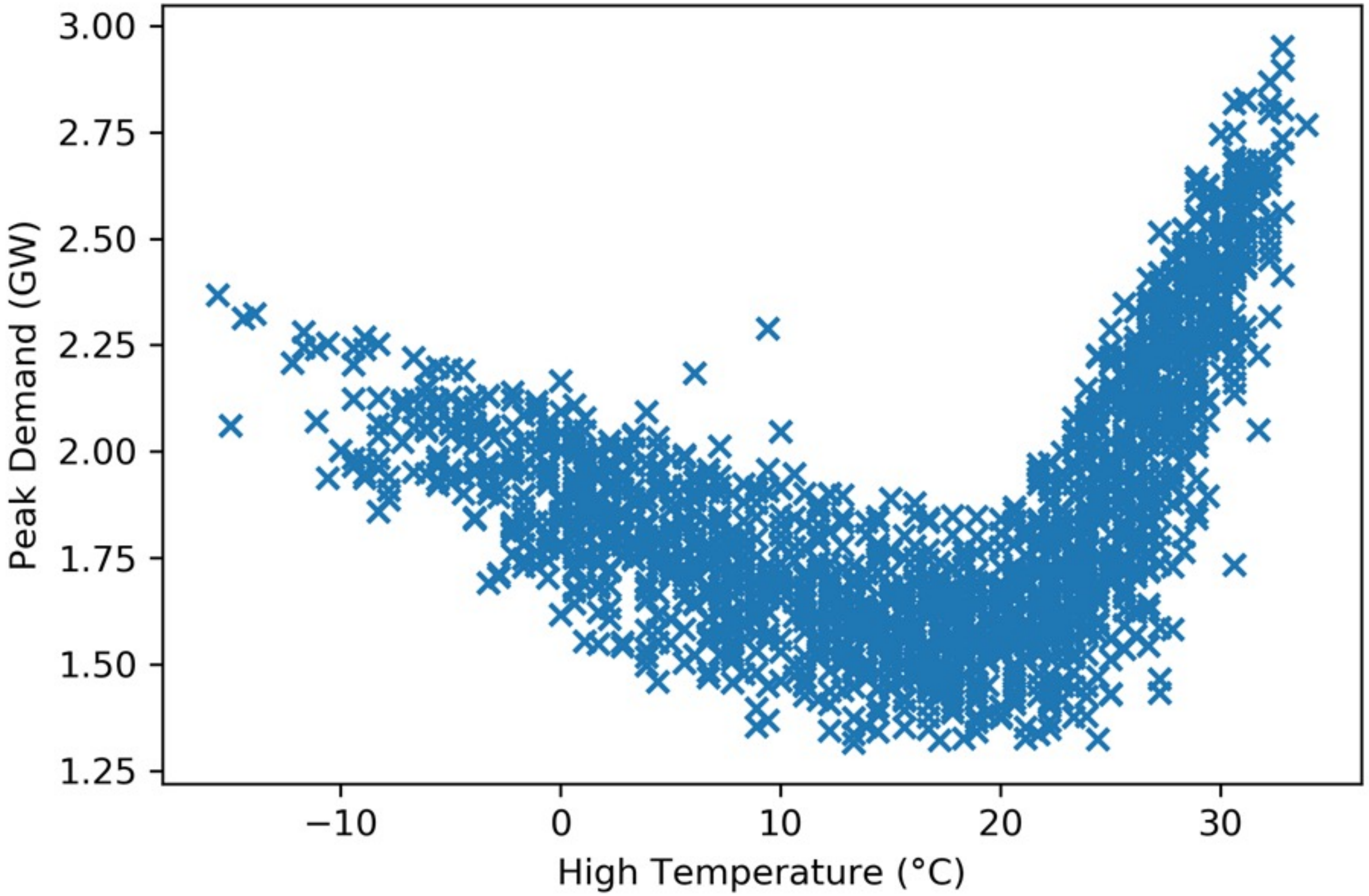
Linear Regression

Beyond Linearity – Polynomial Regression

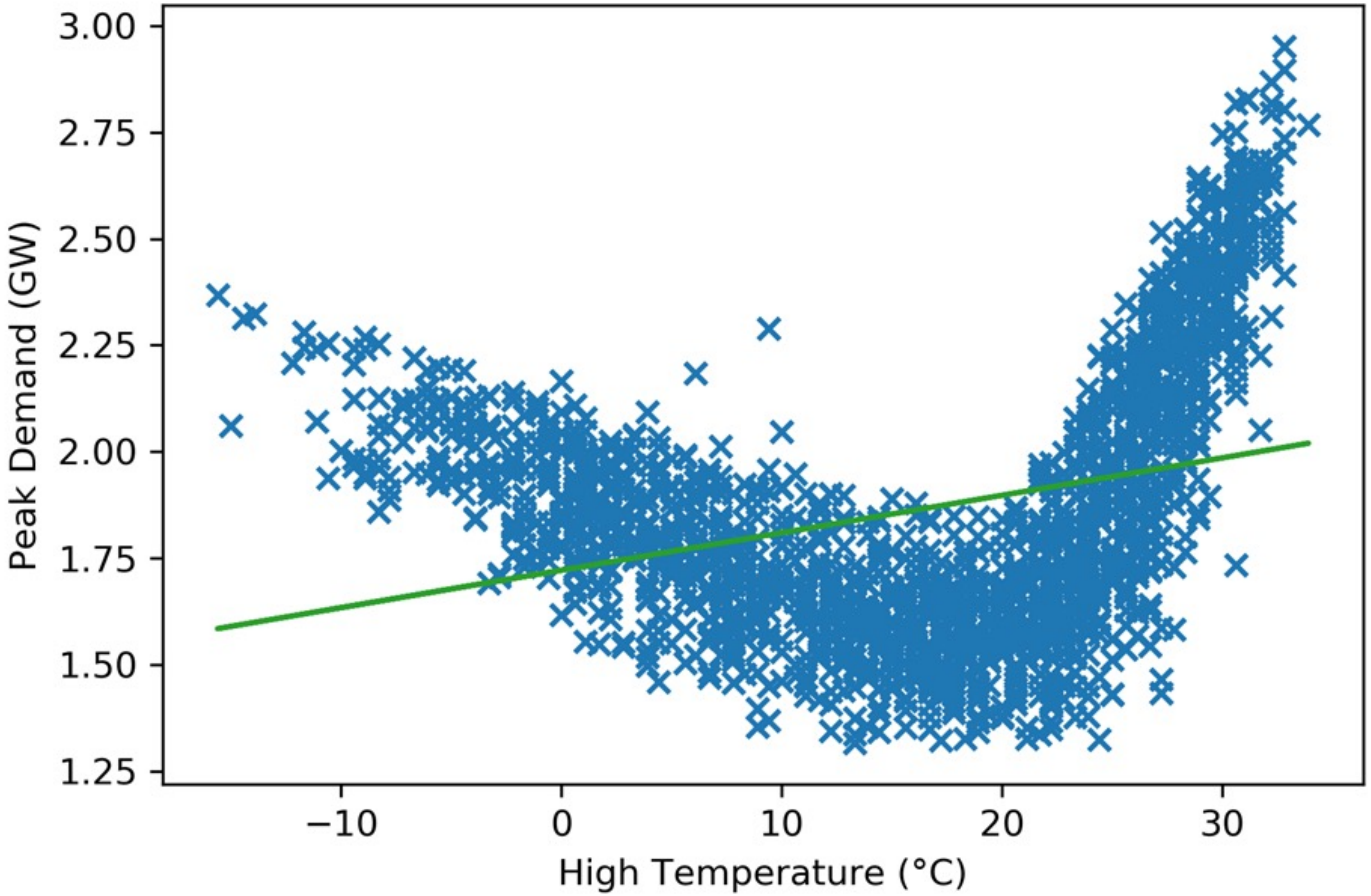
Peak demand vs. high temperature (summer months)



Peak demand vs. high temperature (all months)



Linear regression fit



"Non-linear" regression

- Thus far, we have illustrated linear regression as "drawing a line through through the data", but this was really a function of our input features
- Though it may seem limited, linear regression algorithms are quite powerful when applied to **non-linear relationships** between **input features** and **target features**
- In our electricity consumption example , e.g.

$$x^{(i)} = \begin{bmatrix} (High\_Temperature^{(i)})^2 \\ High\_Temperature^{(i)} \\ 1 \end{bmatrix}$$

Polynomial Regression

	Polynomial Regression of Degree N	Example for N = 3
Input features	$x^{(i)} \in \mathbb{R}^{N+1}, i = 1, \dots, m$	$x^{(i)} = \begin{bmatrix} 1 \\ \dots \\ (High\_Temperature^{(i)})^3 \end{bmatrix} = \begin{bmatrix} 1 \\ \dots \\ (x_1^{(i)})^3 \end{bmatrix}$
Target features	$y^{(i)} \in \mathcal{Y} \subseteq \mathbb{R}, i = 1, \dots, m$	$y^{(i)} \in \mathbb{R} = Peak\_Demand^{(i)}$
Model parameters	$\theta \in \mathbb{R}^{N+1}, \theta = (\theta_0, \dots, \theta_N)$	$\theta = (\theta_0, \dots, \theta_N)$
Hypothesis function	$h_\theta : \mathbb{R} \to y,$ $\quad h_\theta(x) = \sum_{j=0}^{N} \theta_j . x_j = \sum_{j=0}^{N} \theta_j \, (x_1)^j$	$\hat{y}^{(i)} := h_\theta\left(x_1^{(i)}\right) = \theta_0 + \theta_1 \, (x_1^{(i)})^1 + \theta_2 (x_1^{(i)})^2 + \theta_3 \, (x_1^{(i)})^3$
Objective function	$\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$	$\ell(\hat{y}, y) = (\hat{y} - y)^2$

Let us re-visit the matrix notation

- Our “data” matrix X with m observations then becomes

$$X = \begin{pmatrix} 1 & \cdots & (x^{(1)})^N \\ \vdots & \ddots & \vdots \\ 1 & \cdots & (x^{(m)})^N \end{pmatrix}$$

- The response vector Y

$$Y = \begin{pmatrix} Y_0 \\ \dots \\ Y_m \end{pmatrix}$$

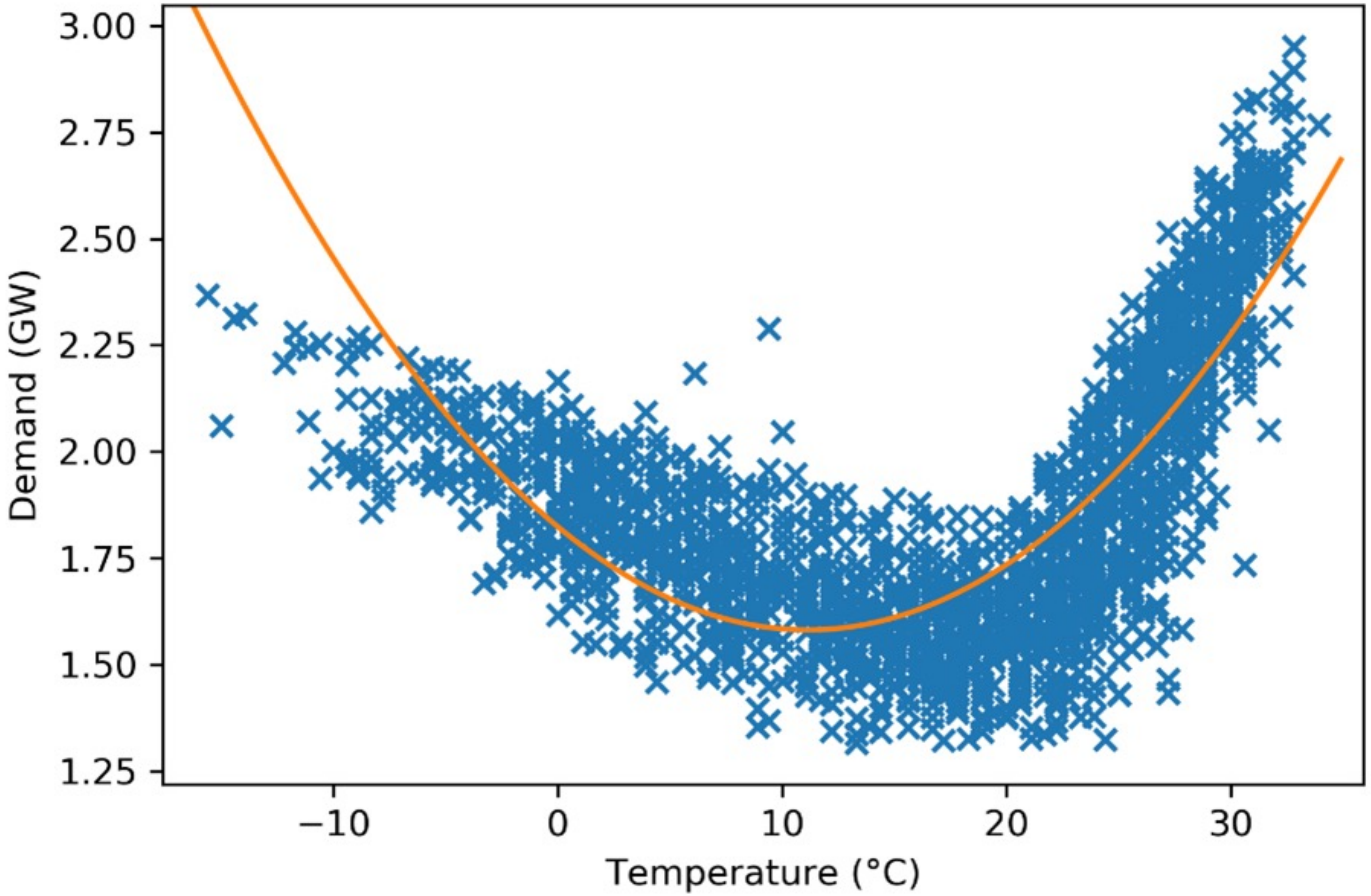
- The parameter vector θ

$$\theta = \begin{pmatrix} \theta_0 \\ \dots \\ \theta_N \end{pmatrix}$$

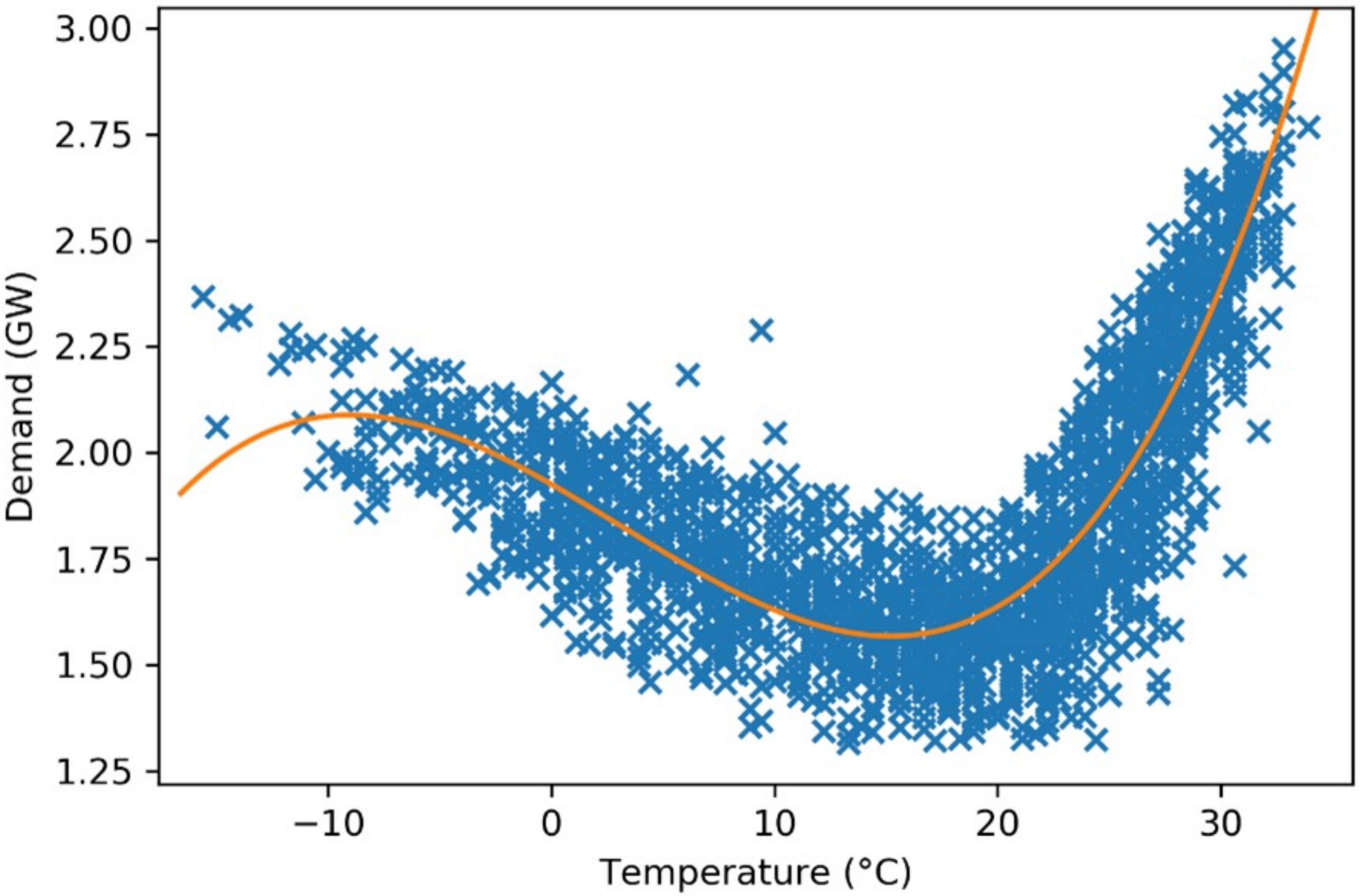
- Thus, we can also use the normal equation to obtain the optimal parameters

$$\theta^* = (X^T X)^{-1} X^T Y$$

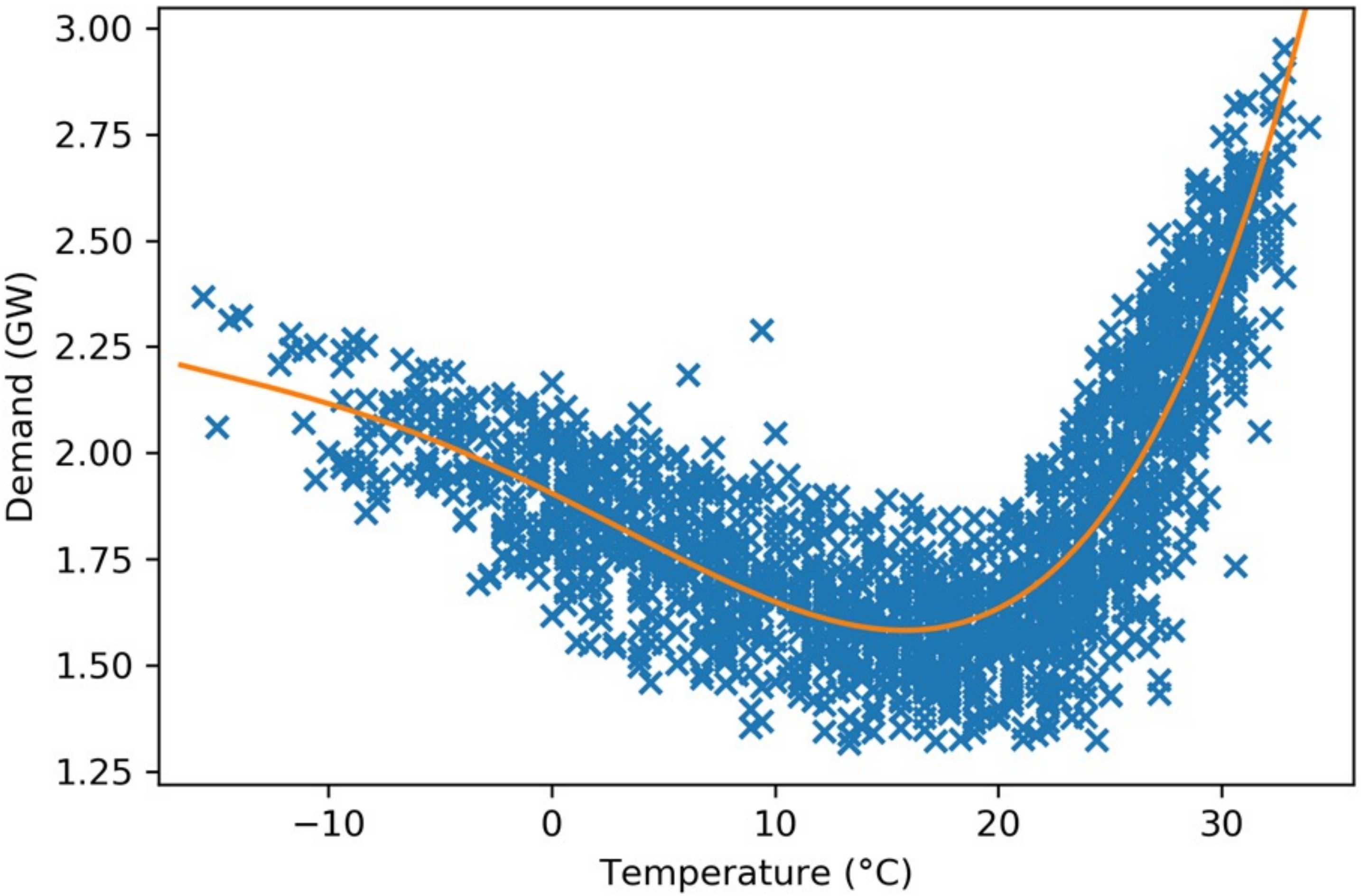
Polynomial features of degree 2 (Peak demand)



Polynomial features of degree 3 (Peak demand)



Polynomial features of degree 4 (Peak demand)



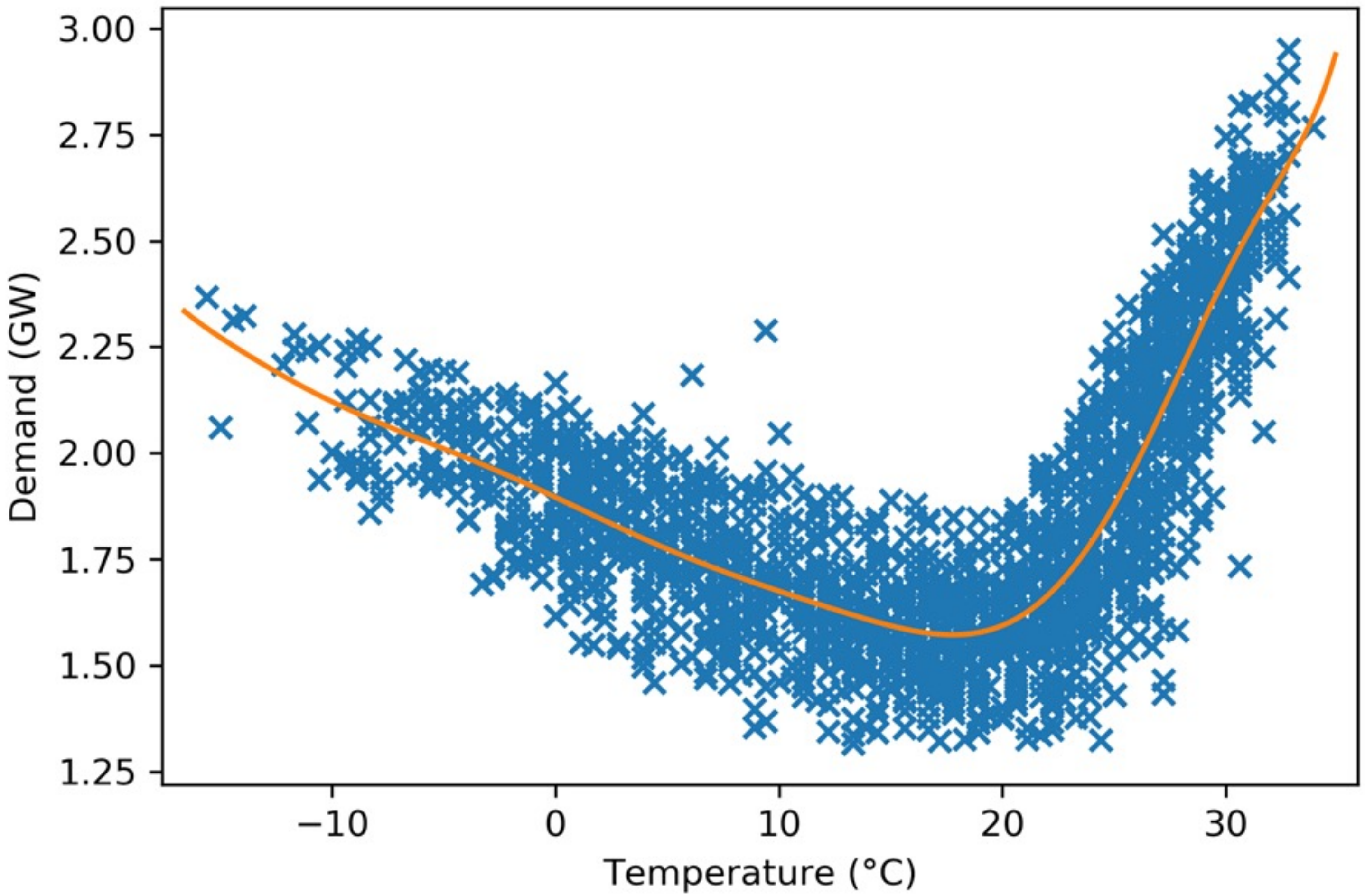
Poll: polynomial regression models

How would the feature vector for a **5-degree polynomial regression** look like for the electricity consumption example from above?

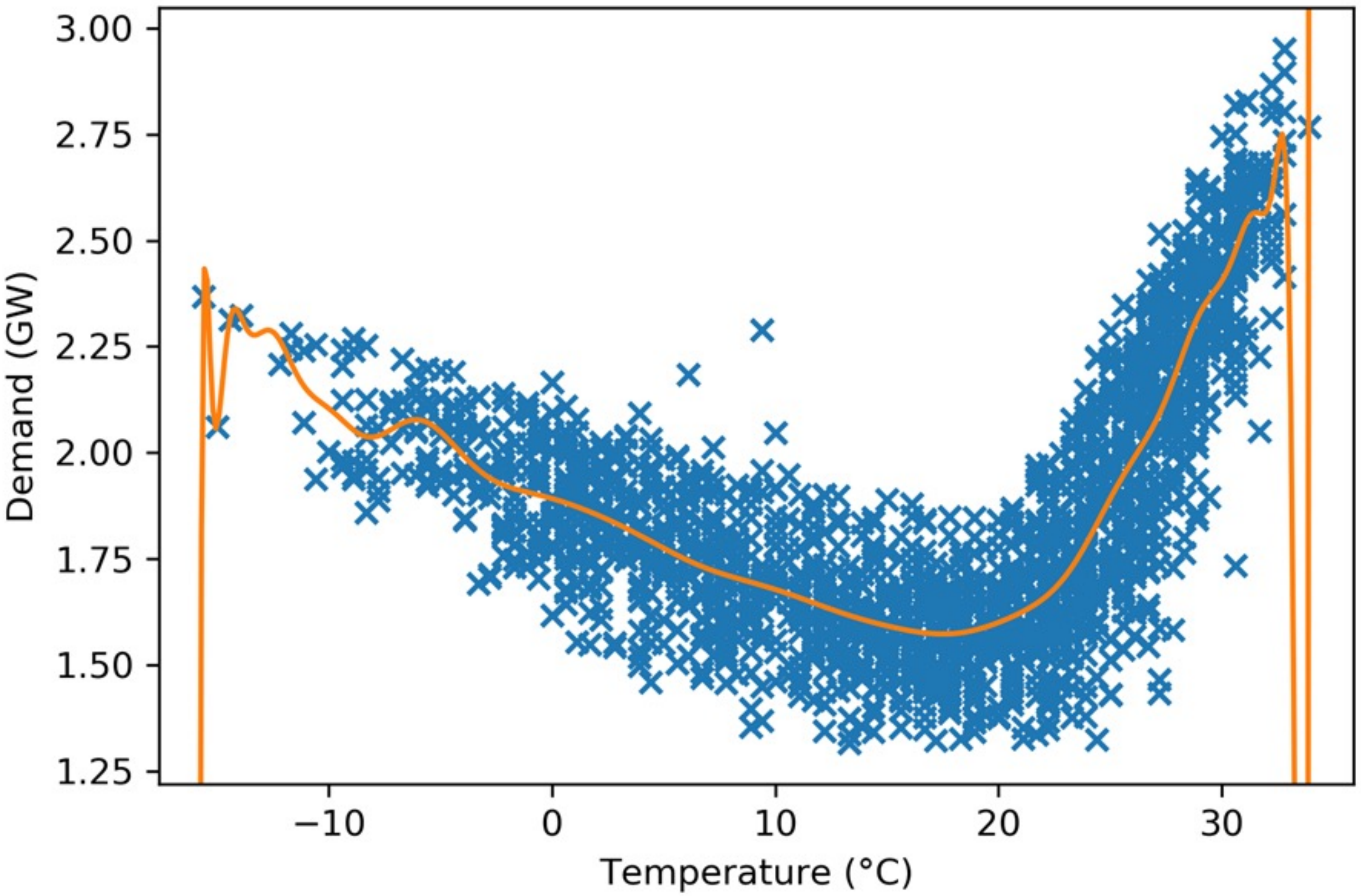
Poll: polynomial regression models

$$x^{(i)} = \begin{bmatrix} 1 \\ (High\_Temperature^{(i)})^1 \\ (High\_Temperature^{(i)})^2 \\ (High\_Temperature^{(i)})^3 \\ (High\_Temperature^{(i)})^4 \\ (High\_Temperature^{(i)})^5 \end{bmatrix}$$

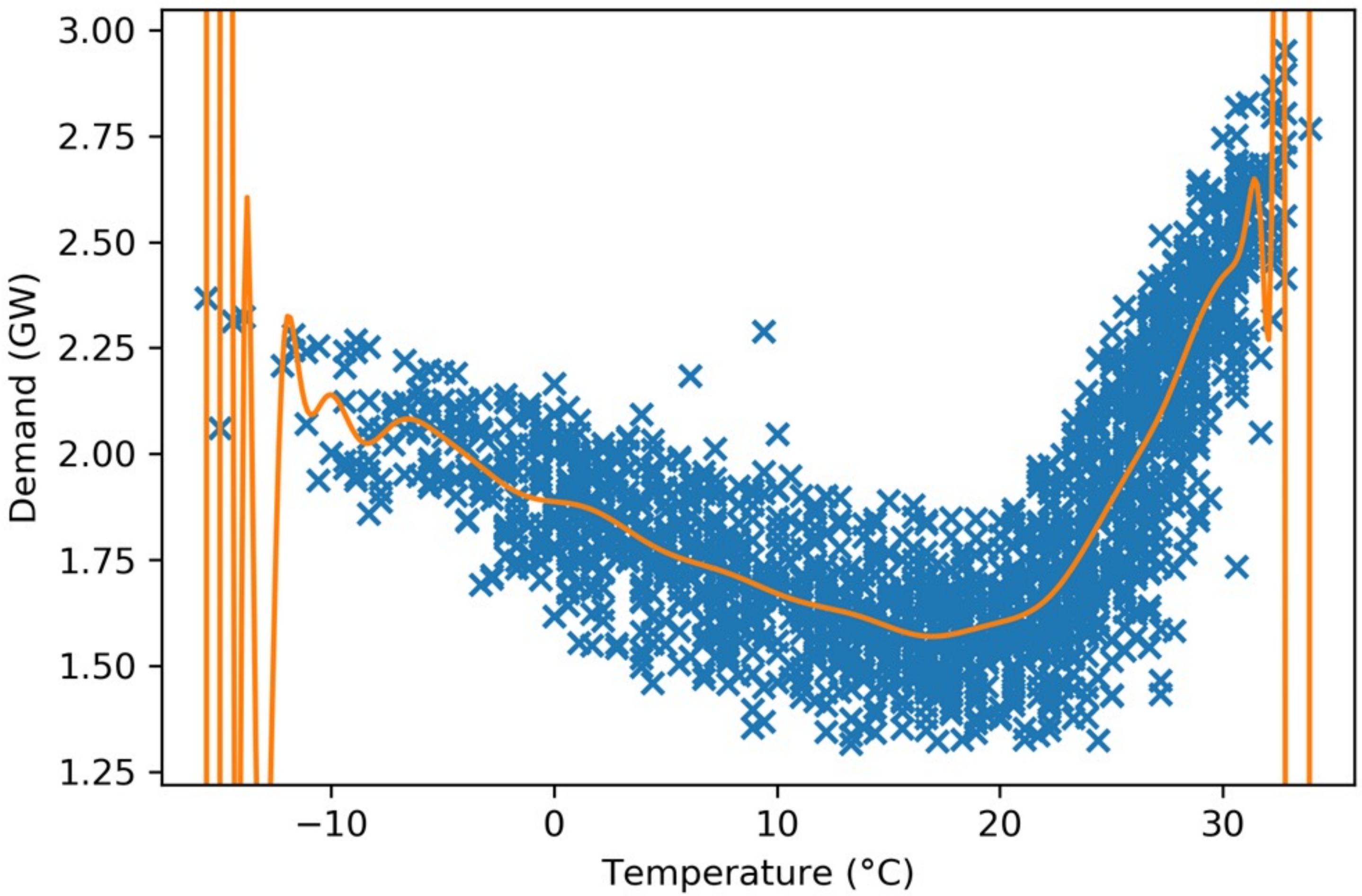
Polynomial features of degree 10 (Peak demand)



Polynomial features of degree 50 (Peak demand)



Polynomial features of degree 100



What do you think has happened with the high polynomials? What are your assumptions on predictive performance?

a) The model fits the data perfectly, thus I am expecting high predictive performance
b) The perfect fit on the data might not generalize well to new observations, thus predictive performance is low
c) Due to the perfect fit, predictive performance is mediocre working well in some cases and not so well in others

What do you think has happened with the high polynomials? What are your assumptions on predictive performance?

a) The model fits the data perfectly, thus I am expecting high predictive performance

b) The perfect fit on the data might not generalize well to new observations, thus predictive performance is low

c) Due to the perfect fit, predictive performance is mediocre working well in some cases and not so well in others

Contact

For general questions and enquiries on **research**, **teaching**, **job openings** and new **projects** refer to our website at www.is3.uni-koeln.de

For specific enquiries regarding this course contact us by sending an email to the **IS3 teaching** address at is3-teaching@wiso.uni-koeln.de