ERROR-BASED LEARNING:

REGRESSION

CS576 MACHINE LEARNING

Dr. Jin S. Yoo, Professor
Department of Computer Science
Purdue University Fort Wayne

Reference

- Kelleher et al., Fundamentals of Machine Learning for Predictive Data Analytics, 2nd edition.
  - Ch 7. Error-Based Learning
  - Appendix C. Differentiation Techniques for Machine Learning
- James et al., An Introduction to Statistical Learning, Ch 3.1-3.2

Outline: Part I

- Fundamentals
  - General Idea for Error-based Learning
  - Simple Linear Regression
  - Measuring Error
  - Error Surfaces
- Standard Approach: Multivariate Linear Regression with Gradient Descent
  - Multivariate Linear Regression
  - Gradient Descent
  - Choosing Learning Rates & Initial Weights
  - A Worked Example

Outline: Part II

- Extensions and Variations
  - Interpreting Multivariable Linear Regression Models
  - Assessing the Parameter Estimates
  - Setting the Learning Rate Using Weight Decay
  - Handling Categorical Descriptive Features
  - Handling Categorical Target Features: Logistic Regression
  - Modeling Non-linear Relationships
  - Multinomial Logistic Regression

General Idea for Error-based Learning

- In a family of error-based machine learning algorithms,
  - **a parameterized prediction model** is **initialized with a set of random parameters**,
  - an **error function** is used to judge how well this initial model performs when making predictions for instances in a training dataset, and
  - based on the value of the error function **the parameters are iteratively adjusted** to create a more and more accurate model.

Outline

- Fundamentals
  - General Idea for Error-based Learning
  - ☞ **Simple Linear Regression**
  - Measuring Error
  - Error Surfaces
- Standard Approach: Multivariate Linear Regression with Gradient Descent
- Extensions and Variations

Regression

- Given a training set of examples:$\{\mathrm{x}_i, y_i\}_{i=1}^{N}$, where $y_i \in \mathbb{R}$ and $\mathrm{x}_i \in \mathbb{R}^D$,

  in **regression** within the context of machine learning,

  we are aiming to **develop a prediction model represented by an unknown function $\boldsymbol{f(x)}$, so that, for an input $\boldsymbol{x_i}$, the output $\boldsymbol{y_i}$ is such that $\boldsymbol{y_i = f(x_i)}$, $\boldsymbol{f(x_i) \in \mathbb{R}}$**

- Our objective is to make accurate prediction for the dependent (target) feature y_i, based on the independent (descriptive) features x_i. The function $f(\mathrm{x}_i)$ is utilized to generate predictions.

Example Dataset

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

Table. **Office rentals dataset:** A dataset that includes office rental prices and a number of descriptive features for 10 Dublin city- center offices.

Relationship between Two Continuous Features

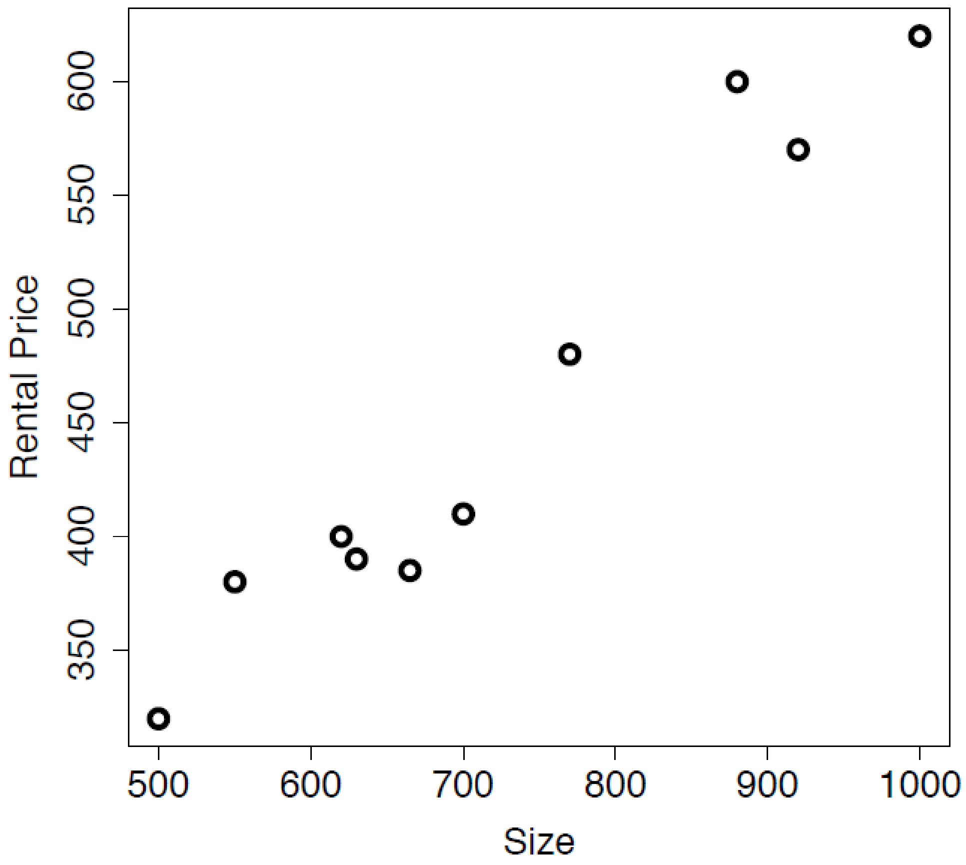


Figure: A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset

- **A strong linear relationship** between SIZE and RENTAL PRICE features
- If we could capture this relationship in a model,
  - First, we would be able to understand how office size affects office rental price.
  - Second, we would be able to fill in the gaps in the dataset to predict office rental prices for office sizes that we have never actually seen in the historical data, e.g., how much would we expect a 730-square-foot office to rent for?

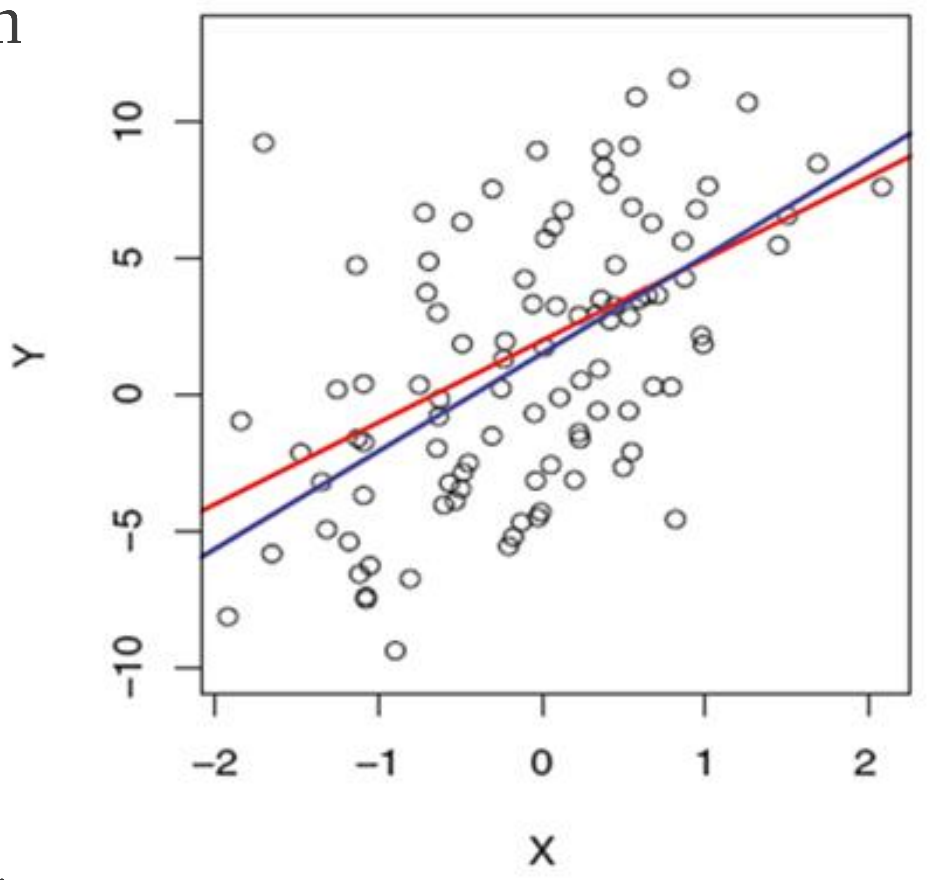
Simple Linear Regression

- A **simple linear regression** is a well-known mathematical model that can capture a linear relationship between two continuous (quantitative) variable.
- A linear relationship (**equation of a line**) between these two variables X and Y as:

$$\boldsymbol{Y = \beta_0 + \beta_1 X + \epsilon}, \qquad \text{(i.e., } Y \approx \beta_0 + \beta_1 X\text{)}$$

, where $\boldsymbol{\beta_0}$ and $\boldsymbol{\beta_1}$ (known as ***coefficients*** or ***parameters*** or *feature-touching* ***weights***) are two unknown constants

  - X is regarded as the predictor, explanatory, or independent variable
  - Y is the response, outcome, or dependent variable
  - $\boldsymbol{\beta_0}$ represents the ***intercept*** of the line, that is, the expected value of Y when X=0
  - $\boldsymbol{\beta_1}$ represent the ***slope*** – the average increase in Y associated with a one-unit increase in X
  - $\boldsymbol{\epsilon}$ is the ***error*** term for what we miss with the model $\beta_0 + \beta_1 X$.



Example: Simple Linear Regression

- The simple linear regression model for SIZE and RENTAL is:

$$\textbf{RENTAL PRICE} = \mathbf{6.47 + 0.62 \times} \textbf{ SIZE}$$

- Using this model, determine the expected rental price of the 730 square foot office.

$$\text{RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE}$$

$$\text{RENTAL PRICE} = 6.47 + 0.62 \times 730$$

$$= 459.07$$

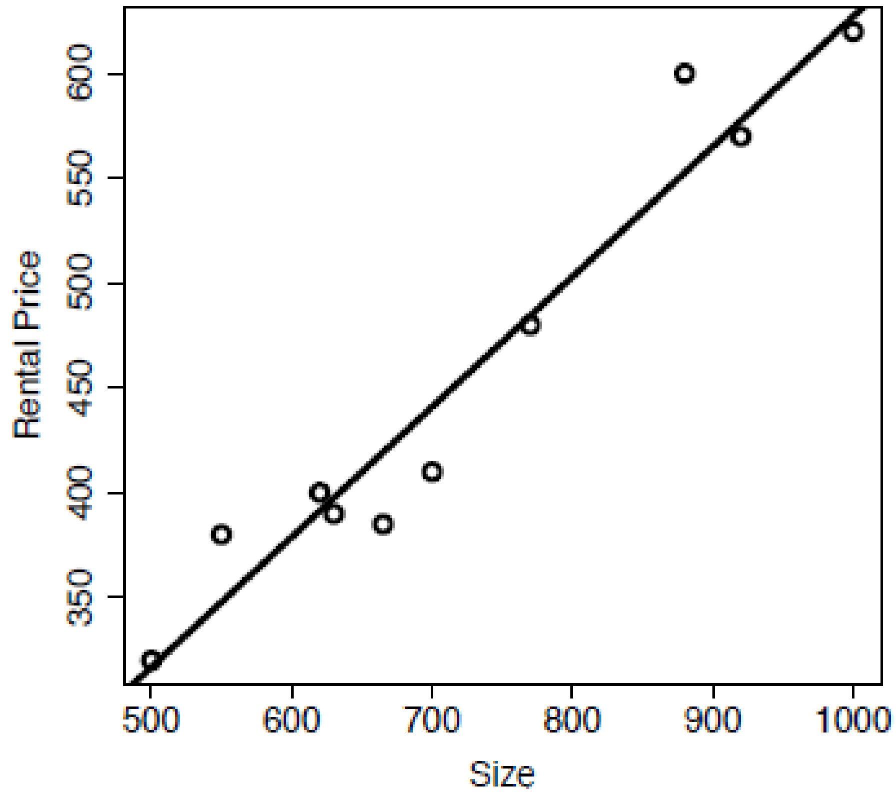


Figure A scatter plot of the SIZE and RENTAL PRICE features with a linear model relating RENTAL PRICE to SIZE overlaid.

Exploring Linear Regression: A Practical Approach

- Despite its simplicity, linear regression is a powerful tool in conceptual understanding and real-world applications, and it continues to be a widely used technique.
- With advertising data, linear regression helps us address several relevant questions, such as:
  - **Exploring Relationships**: Is there a discernible relationship between advertising budget and sales? If so, Is the relationship linear?
  - **Measuring Strength**: How strong is the connection between advertising budget and sales?
  - **Predictive Accuracy**: How precisely can we forecast future sales using these relationships?
  - **Assessing Contribution**: Can we identify which media types are pivotal contributors to sales?
  - **Synergistic Analysis**: Is there observable synergy among different advertising media affecting sales?

Supervised Learning - Simple Linear Regression

- We will formally describe the **supervised learning method** called ***linear regression*** or the fitting of a representative *line* (or, in higher dimensions, a *hyperplane*) to a set of data points
- We **assume there is approximately a linear relationship** between the descriptive feature and the target feature,
- The **Simple Linear Regression** model using a single descriptive feature is formally described as

$$\mathbb{M}_{w}(\mathbf{d}) = \mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}[1]$$

, where **w** is the vector of weights <**w**[0], **w**[1]>, **d** is an instance by a single descriptive feature **d**[1], and $\mathbb{M}_{w}(\mathbf{d})$ is the prediction output by the model for the instance **d**

Simple Linear Regression

- Simple linear regression involves fitting a ***line*** through the scattered data points in a two-dimensional space.
- The challenge lies in determining the optimal values for the weights **w**[0] and **w**[1] that best represent the relationship between the descriptive features and the target feature.
- **Example**: Among candidate models, which one most accurately fits the relationship between office sizes and office rental prices?
  - The third one (line) from the top (with **w**[1] set to 0.62)

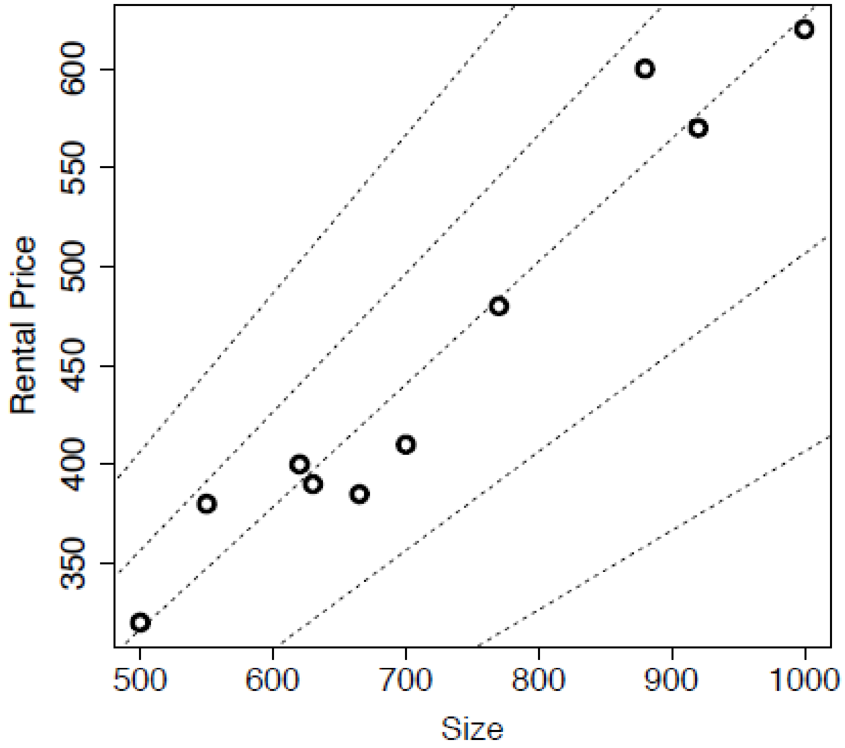


Figure: A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset, and a collection of possible simple linear regression models capturing the relationship between these two features. For all models **w**[0] is set to 6.47. From top to bottom the models use 0.4, 0.5, 0.62, 0.7 and 0.8 respectively for **w**[1].

Errors

- What is the way **to measure how well a model** defined using a candidate set of weights fits a training dataset ?
- We measure the **error** (or **residual**) between the predictions made by a model and the actual values t in a training dataset.

  In fact, $t = \mathbb{M}_w(\mathrm{d}) + \epsilon$

  $$= \mathrm{w}[0] + \mathrm{w}[1] \times \mathrm{d}[1] + \epsilon$$

- An **error function** is used to formally measure the fit of a linear regression with training data.

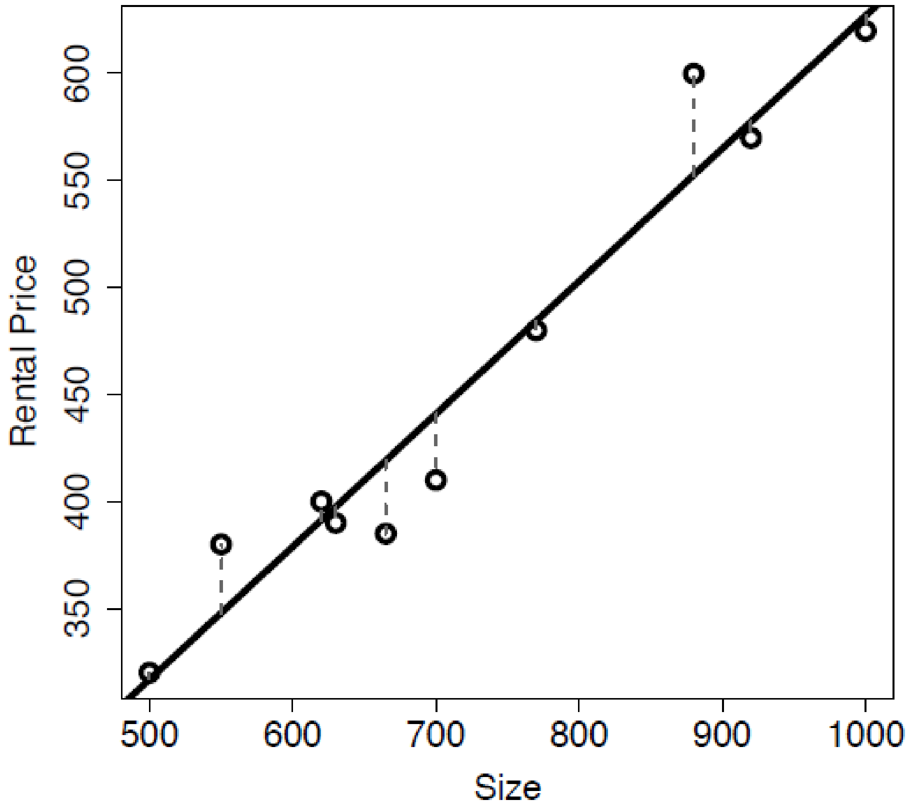


Figure: A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset showing a prediction model (with **w**[0] = 6.47 and **w**[1] = 0.62) with **the resulting errors.**

Measuring Errors: Sum of Squared Errors

- There are a number of ways of measuring ***closeness*** (different kinds of error functions) for linear regression models
- The most common one is the **Sum of Squared Error** (SSE), or $\boldsymbol{L_2}$ (*Euclidean norm*), (also referred to ***Least Squares*** cost function)

$$\boldsymbol{SSE}(\mathbb{M}_w, \mathcal{D}) = \boldsymbol{L_2}(\mathbb{M}_w, \mathcal{D}) = \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{t_i} - \mathbb{M}_w(\mathbf{d}_i[1]))^2$$

$$= \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{t_i} - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2$$

$$= \frac{1}{2}((t_1 - \hat{t}_1)^2 + (t_2 - \hat{t}_2)^2 \ldots + (t_n - \hat{t}_n)^2)$$

, where n training instances, each instance is composed of a single descriptive feature $\mathbf{d}[1]$ and a target feature t .
$\mathbb{M}_w(\mathbf{d}_i)$ is the prediction made by a candidate model $\mathbb{M}_w$ for a training instance with descriptive features $\mathbf{d}_i[1]$.

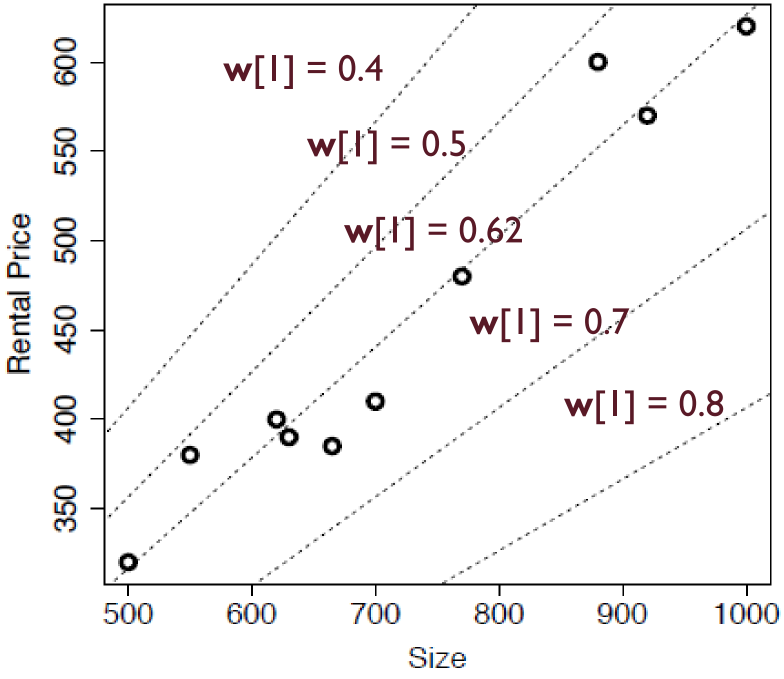
- **NOTE**: There are slightly **different SSE equations** (also called RSS (residual sum of squares) or RSE (residual standard error)) **but the same for measuring the errors**):

$$\sum_{i=1}^{n}(t_i - \hat{t}_i)^2 \text{ , } \frac{1}{n}\sum_{i=1}^{n}(t_i - \hat{t}_i)^2, \frac{1}{n-2}\sum_{i=1}^{n}(t_i - \hat{t}_i)^2, \frac{1}{2n}\sum_{i=1}^{n}(t_i - \hat{t}_i)^2, \text{ and } \frac{1}{2}\sum_{i=1}^{n}(t_i - \hat{t}_i)^2$$

Example: Measuring Errors

- RENTAL PRICE
  $= 6.47 + w[1] \times \text{SIZE}$

  when **w**[0] is fixed at 6.47 and
  - **w**[1] = 0.4, *SSE* is 136,218
  - **w**[1] = 0.5, *SSE* is 42,712
  - **w[1] = 0.62, *SSE* is 2835.82**
  - **w**[1] = 0.7, *SSE* is 20,092
  - **w**[1] = 0.8, *SSE* is 90,978



- **Example**: Calculating the sum of squared errors for the candidate model (with **w**[0] = 6.47 and **w**[1] **= 0.62**) making predictions for the office rentals dataset.

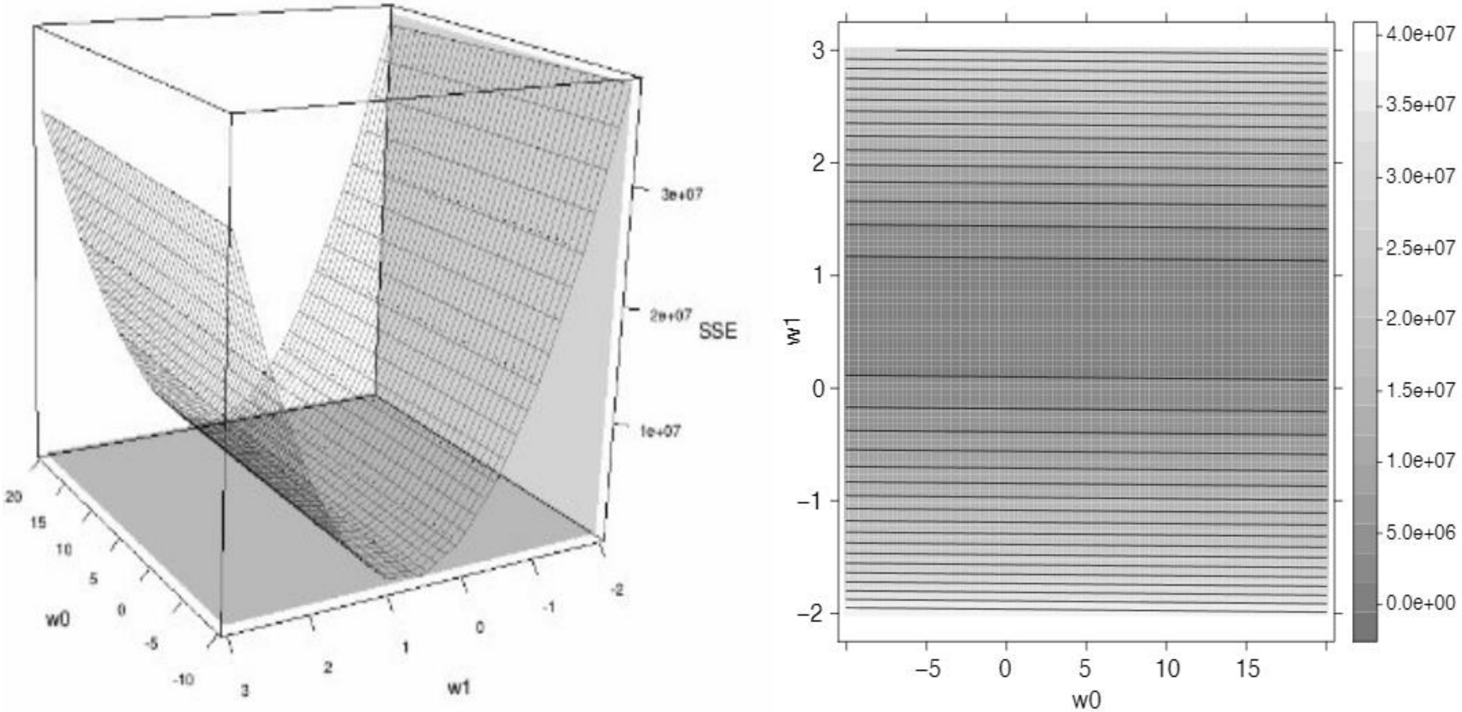
ID	RENTAL PRICE	Model Prediction	Error	Squared Error
1	320	316.79	3.21	10.32
2	380	347.82	32.18	1,035.62
3	400	391.26	8.74	76.32
4	390	397.47	-7.47	55.80
5	385	419.19	-34.19	1,169.13
6	410	440.91	-30.91	955.73
7	480	484.36	-4.36	19.01
8	600	552.63	47.37	2,243.90
9	570	577.46	-7.46	55.59
10	620	627.11	-7.11	50.51
			Sum	**5,671.64**
	Sum of squared errors (Sum/2)			**2,835.82**

- **The best model with lowest SSE** is

  $\text{RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE}$

Error Surface

- For each possible combination of weight parameters, a corresponding value of sum of squared errors (SSE) can be computed, forming a **surface** when these values are connected together.
  - Each pair of weight **w**[0] and **w**[1] defines a point on the *x*-*y* plane which is known as a **weight space**
  - The sum of squared errors for the model using these weights determines the height of the **error surface** above the *x*-*y* plane for that pair of weights.



(a) (b)

Figure: (a) A **3D surface plot** and (b) **a contour plot** of the error surface generated by plotting the sum of squared errors (SSE) value for the office rentals training set for each possible combination of values for **w**[0] (from the range [-10, 20]) and **w**[1] (from the range [-2, 3]).

Properties of Error Surface

- **Two properties of error surfaces**: **convex** and **a global minimum**.
  - The SSE cost function for linear regression is to be ***convex*** for any dataset. By convex the error surfaces are shaped like a **bowl**.
  - **Having a global minimum** means that on an error surface, **there is a unique set of optimal parameters with the lowest SSE.**
- So, **the best-fit set of weights for a linear regression model can be found at the global minimum of the error surface** (the lowest point on the error surface) defined by the weight space associated with the relevant training dataset.
- This approach to finding weights is known as **least squares optimization**.
  - The linear regression which uses least squares for the error function is called ***Least Squares Linear Regression***

Model Fitting – Global Minimum for Optimal Weights

- The global minimum can be found at the point at which the **partial derivatives** of the error surface, with respect to the weights **w**[0] and **w**[1], **are equal to 0.**

$$\frac{\partial}{\partial \mathbf{w}[0]} \frac{1}{2} \sum_{i=1}^{n} (t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 = 0$$

and

$$\frac{\partial}{\partial \mathbf{w}[1]} \frac{1}{2} \sum_{i=1}^{n} (t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 = 0$$

  - The **partial derivatives** of the error surface with respect to w[0] and w[1] **measure the slope of the error surface** at the point w[0] and w[1]
  - There is <u>no slope</u> at the bottom of the bowl (slope=0).
  - The point on the error surface at which the partial derivatives with respect to w[0] and w[1] are equal to 0 <u>is simply the point at the very bottom</u> of the bowl defined by the error surface.
  - This point is at the global minimum of the error surface

- <u>The coordinates of the bottom point define the weights for the prediction model</u> with the lowest sum of squared errors on the dataset.

Outline

- Fundamentals
- ☞Standard Approach: **Multivariate Linear Regression with Gradient Descent**
  - Multivariate Linear Regression
  - Gradient Descent Algorithm
  - Choosing Learning Rates & Initial Weights
  - A Worked Example
- Extensions and Variations

Multiple Linear Regression

- ***Multiple linear regression*** is an extension of the simple linear regression to include multiple independent variables. It seeks to model the relationship between two or more features and a response by fitting a linear equation to observed data.
- The multiple linear regression model is

$$\boldsymbol{Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_d X_d + \epsilon}$$

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_d X_d$$

, where β_0 is the y-intercept, β_1, ..., β_d are the coefficients of the independent variables, $X_1, X_2, \ldots, X_d$

  - β_j reflects the *average* effect on Y of a one unit increase in X_j , *holding all other predictors fixed.*

- **Example**: A model for an advertising dataset is

$$sales = \beta_0 + \beta_1\, TV + \beta_2\, radio + \ldots + \beta_d\, newspaper + \epsilon$$

Multivariate Linear Regression Model

- The most common approach to error-based machine learning is to use ***multivariate linear regression with gradient descent*** to train a best-fit model for a given training dataset.
- A **multivariate linear regression model** is defined as:

$$
\begin{aligned}
\mathbb{M}_w(\mathbf{d}) &= \mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \cdots + \mathbf{w}[m] \times \mathbf{d}[m] \\
&= \mathbf{w}[0] + \sum_{j=1}^{m} \mathbf{w}[j] \times \mathbf{d}[j] \\
&= \mathbf{w}[0] \times \mathbf{d}[0] + \sum_{j=1}^{m} \mathbf{w}[j] \times \mathbf{d}[j] \quad \text{, where } \mathbf{d}[0] \text{ is a dummy descriptive feature (always 1)} \\
&= \sum_{j=0}^{m} \mathbf{w}[j] \times \mathbf{d}[j] \\
&= \mathbf{w} \cdot \mathbf{d}
\end{aligned}
$$

, where $\mathbf{w} \cdot \mathbf{d}$ is the dot product of the vectors $\mathbf{w}$ and $\mathbf{d}$

  - We interpret $\mathbf{w}[j]$ as the *average* effect on the prediction value of a one unit increase in $\mathbf{d}[j]$, *holding all other descriptive features fixed*.

Example

- The multivariate regression to predict office rental price with size, floor and broadband rate is:

  $\textsc{Rental Price} = \mathbf{w}[0] + \mathbf{w}[1] \times \textsc{Size} + \mathbf{w}[2] \times \textsc{Floor} + \mathbf{w}[3] \times \textsc{Broadband Rate}$

- Suppose the best-fit set of weights found for this equation is:

  w[0] = -0.1513, **w**[1] = 0:6270,
  w[2] = -0.1781, **w**[3] = 0.0714.

- The **prediction model** is :

  $$\textsc{Rental Price} = -0.1513 + 0.6270 \times \textsc{Size} + 0.1781 \times \textsc{Floor} + 0.0714 \times \textsc{Broadband Rate}$$

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

- **Using the model, predict the expected rental price** of a 690 square foot office on the 11th floor of a building with a broadband rate of 50 Mb per second :

  RENTAL PRICE ? $-0.1513 + 0.6270 \times 690 + 0.1781 \times 11 + 0.0714 \times 50 = 434.0896$

Measuring Regression Performance

- The ***Sum of Squared Errors* (SSE)** loss function, for multivariate linear regression $\mathbb{M}_w$ from n training instances is

$$\boldsymbol{SSE} = L_2(\mathbb{M}_w, \mathcal{D}) = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}_w(\mathbf{d}_i))^2 = \frac{1}{2}\sum_{i=1}^{n}(t_i - (\mathbf{w}\cdot\mathbf{d}_i))^2$$

, where defined by the weight vector **w**

- Similarly, the ***Mean Squared Error* (MSE)** $= \frac{1}{n}\sum_{i=1}^{n}(t_i - \mathbb{M}_w(\mathbf{d}_i))^2$
- The quality of a linear regression fit (i.e., the trained model) can be also assessed using the RSE and alternatively, the R^2 statistic.
- ***Residual Standard Error* (RSE)** is an estimate of the standard deviation of $errors$

$$\boldsymbol{RSE} = \sqrt{RSS/(n-2)} = \sqrt{\frac{\sum_{i=1}^{n}(t_i - \mathbb{M}_w(\mathbf{d}_i))^2}{\boldsymbol{n-2}}},$$

where $RSS = \sum_{i=1}^{n}(t_i - \mathbb{M}_w(\mathbf{d}_i))^2$ is the ***residual sum-of-squares* (RSS)**

- ***R-squared* (R^2)** statistic provides the proportion of variability in target value that can be explained using descriptive features. $\mathbf{0} \leq \boldsymbol{R}^{\mathbf{2}} \leq \mathbf{1}$

$$\boldsymbol{R}^{\mathbf{2}} = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where $\mathbf{TSS} = \sum_{i=1}^{n}(y_i - \bar{\bar{y}}_i)^2$ is the ***total sum of squares***

R^2 Statistic

- **An R^2 statistic near 1** signals that the regression model accounts for a large portion of the variance in the target variable.
- Conversely, **an R^2 statistic value close to 0** implies that the regression model does not explain much of the variability in the response.
  - This could occur if the linear model is incorrect, the inherent error is substantial, or both.
- **In the simple linear regression setting, $R^2 = r^2$**, where r is the **correlation** between X and Y:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}.$$

Multivariate Linear Regression – The Least Squares Regression Line

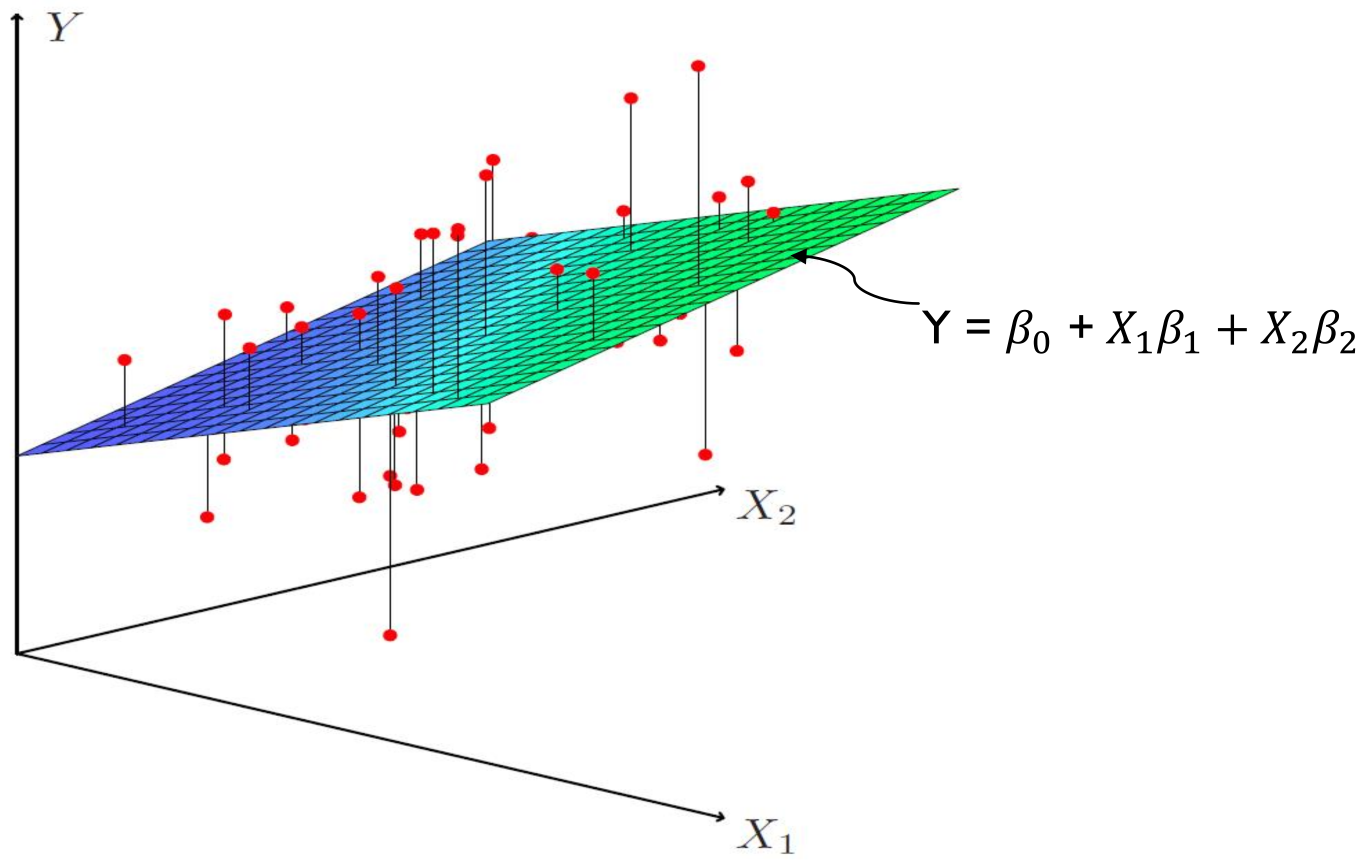


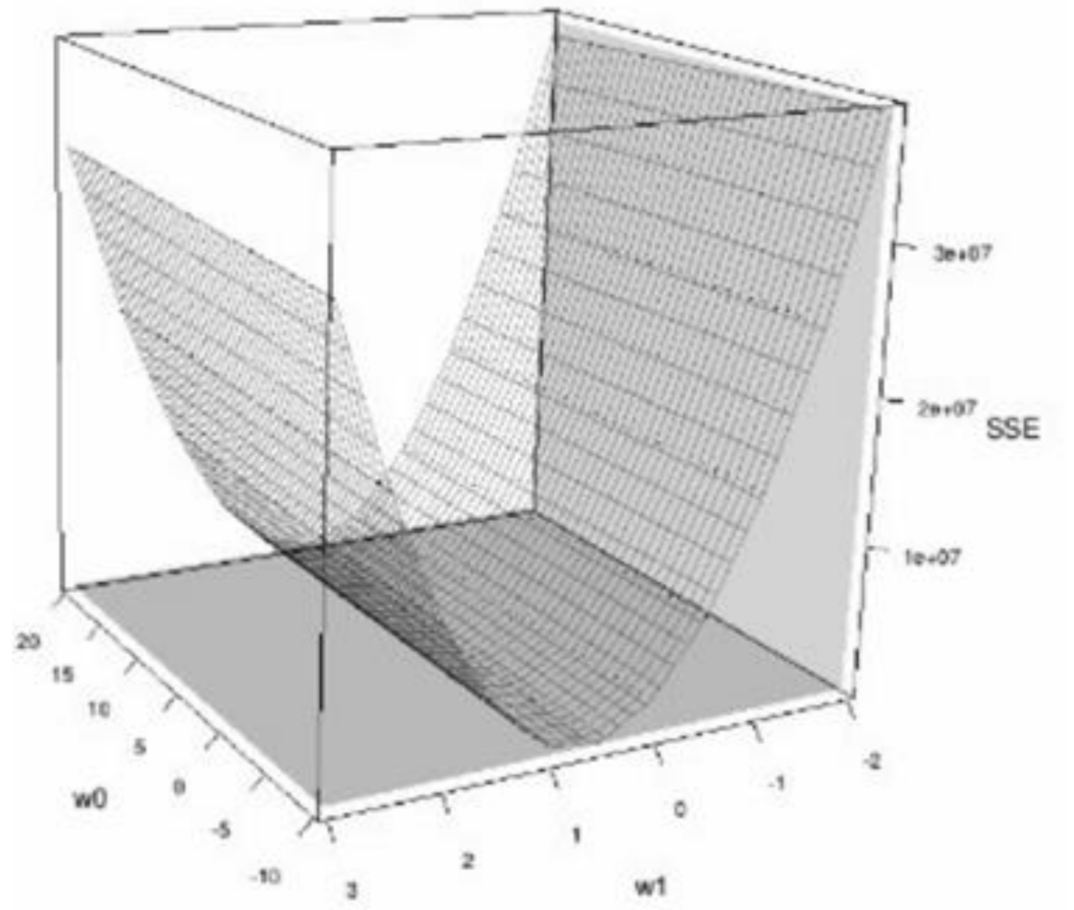
Figure. In a three-dimensional setting, with two independent variables and one response (target), the least squares regression line becomes a **plane**. The plane is chosen to minimize the sum of the squared vertical distances between each observation (shown in red) and the plane.

Outline

- Fundamentals
- Standard Approach: Multivariate Linear Regression with Gradient Descent
  - Multivariate Linear Regression
  - ☞ **Gradient Descent Algorithm**
  - Choosing Learning Rates & Initial Weights
  - A Worked Example
- Extensions and Variations

Model Fitting – Optimal Weight Search

- The **optimal line** of best fit minimizes the error between the predicted values and the actual data points.
- To find the **best-fit weights** for a linear regression model, we locate the global minimum on the error surface, defined by the weight space associated with the relevant training dataset.
- **Identifying the global minimum** is possible at the position where the partial derivatives of the error surface are zero with respect to the weights.

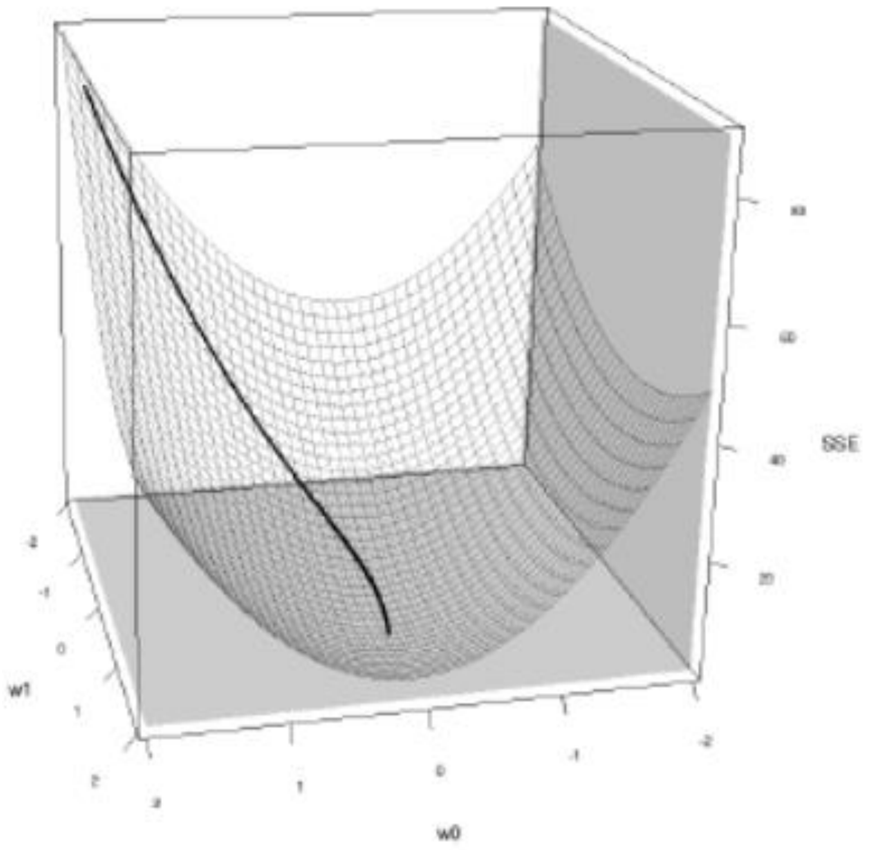


Model Fitting – Optimal Weight Search (cont.)

- **Brute-force search**
  - Test every plausible combination of weight parameters to determine the optimal set of weights.
  - This approach becomes inefficient and impractical as the number of descriptive features, and consequently, the weights, grow.
- Instead, because the error surface (of the convex shape) has a single global minimum, **a guided search approach known as the gradient descent algorithm** is popularly used for finding the best set of weights.

Gradient Descent

- Gradient descent starts by selecting a random point within the weight space (randomly selected weights), and calculating the sum of squared errors associated with this point. This defines one point on the error surface.
- The current location in the weight space is adjusted slightly in the direction in which the slopes most steeply downward.
  - The direction and magnitude of the adjustment to be made to a weight is determined by the gradient of the error surface at the current position in the weight space.
  - The algorithm can use only very localized information. However, it can use the direction of the slope of the error surface (slope up or down) at the current location in the weight space.
- This adjustment is repeated over and over until the global minimum on the error surface



Gradient Descent

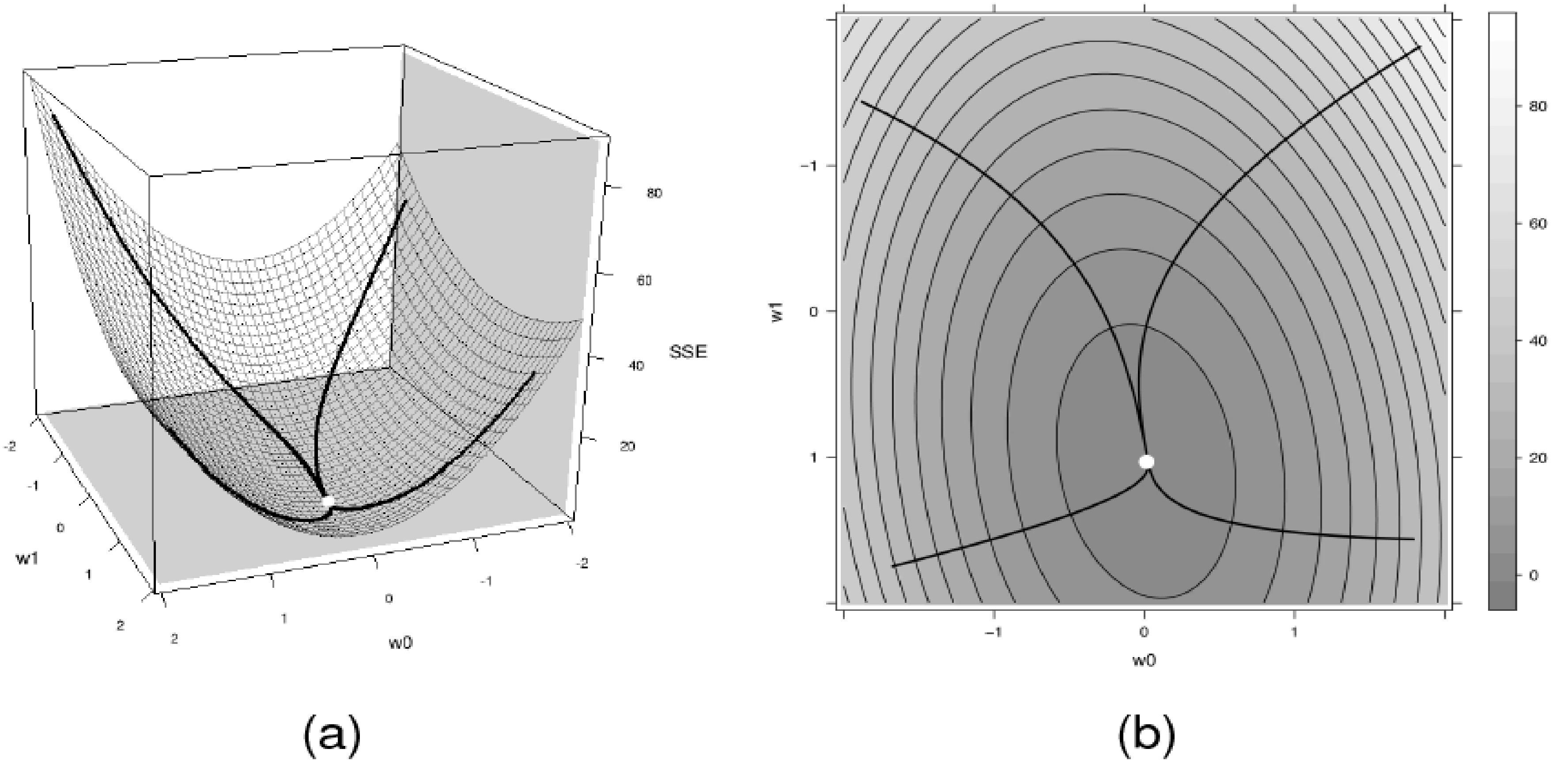


Figure: (a) A 3D surface plot and (b) a contour plot of the same error surface. The lines indicate the path that the gradient decent algorithm would take across this error surface from different starting positions to the global minimum - marked as the white dot in the center.

Example with Office Rentals Data

The journey across the error surface that is taken by the gradient descent algorithm when training the simple version of the office rentals example - involving just Size and Rental Price.

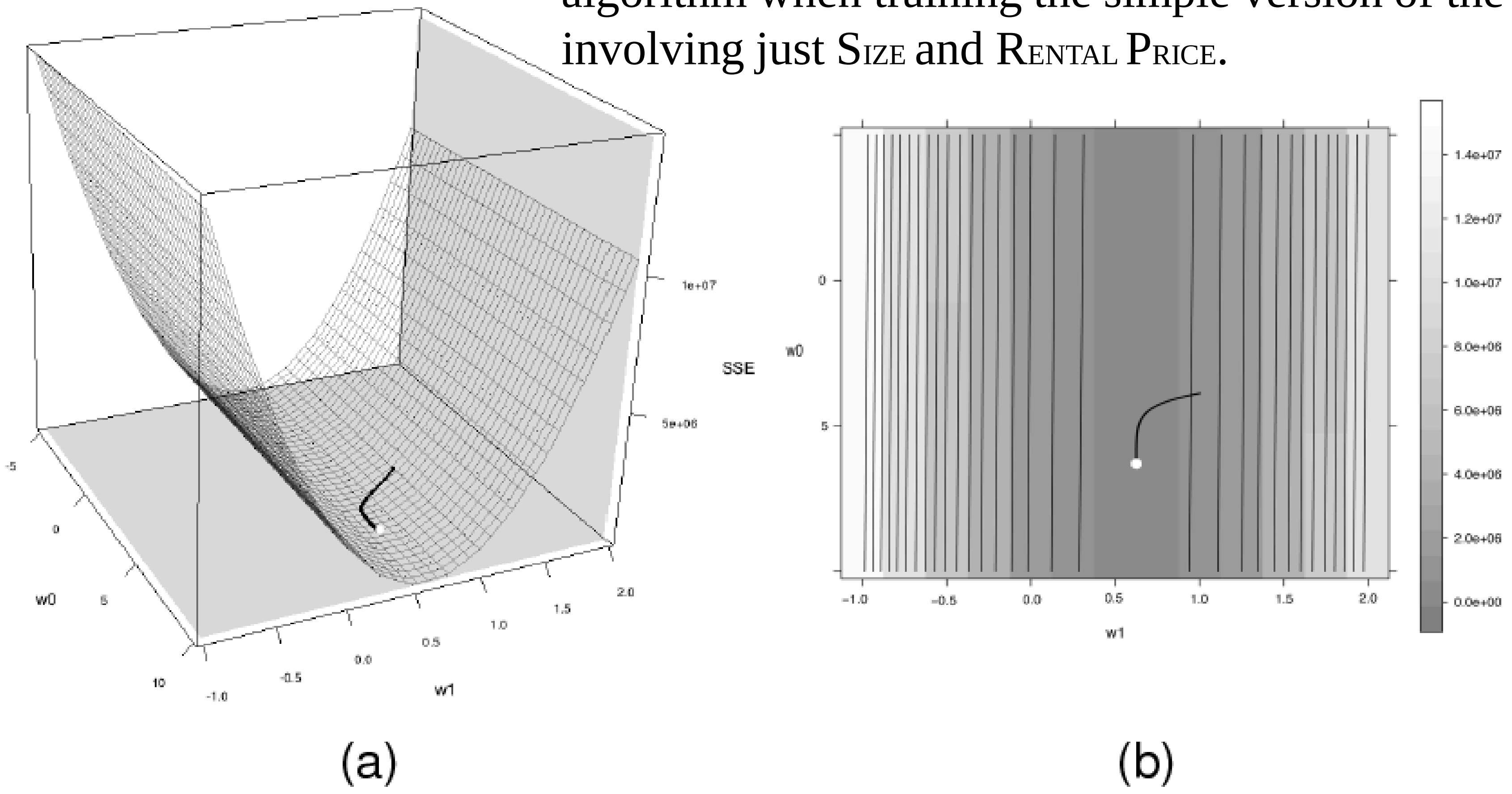


Figure: (a) A 3D surface plot and (b) a contour plot of the error surface for the office rentals dataset showing the path that the gradient descent algorithm takes towards the best fit model.

Example (cont.)

Notice how the model gets closer and closer to a model that accurately captures the relationship between SIZE and RENTAL PRICE.

This is also apparent in the final panel, which shows how the sum of squared errors decreases as the model becomes more accurate.

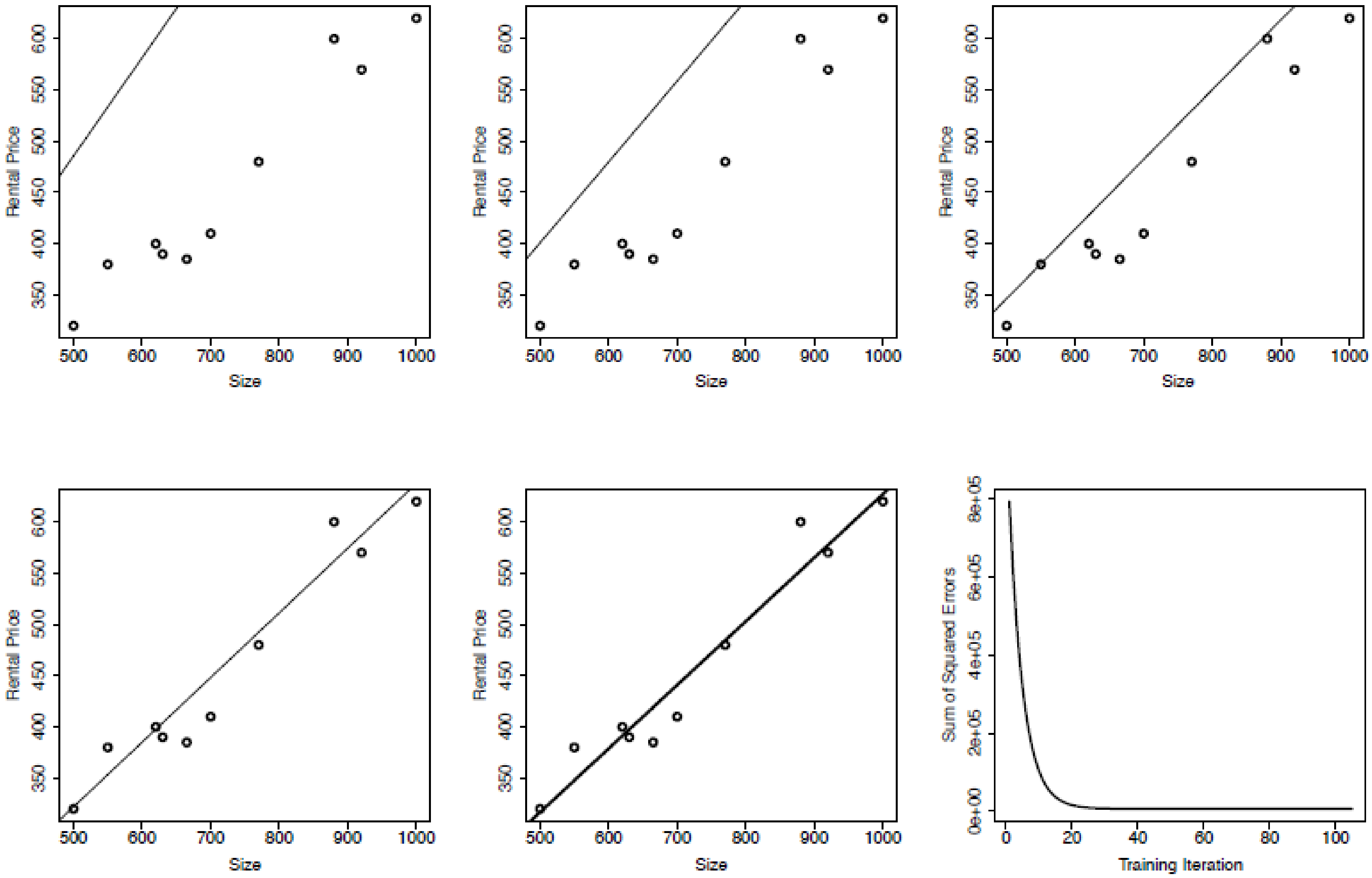


Figure: A series of the simple linear regression models developed during the gradient descent process for the office rentals dataset. The final panel shows the sum of squared error values generated during the gradient descent process.

Gradient Descent Algorithm

- **Input**
  - a set of training examples $\mathcal{D}$
  - a **learning rate** α that controls how quickly the algorithm converges
  - a **errorDelta** function that determines the direction in which to adjust a given weight, **w**[j], so as to move down the slope of an error surface determined by the dataset, $\mathcal{D}$
  - a **convergence criterion** that indicates that the algorithm has completed

$\mathbf{w} \leftarrow$ random starting point in the weight space
repeat
    for each $\mathbf{w}[j]$ in $\mathbf{w}$ **do**
        $\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \mathbf{errorDelta}(\mathcal{D}, \mathbf{w}[j])$
    end for
until convergence occurs

Weight Update

- The key component of the gradient descent algorithm is the **weight update step**:

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \mathbf{errorDelta}(\mathcal{D}, \mathbf{w}[j])$$

  - Each weight is considered independently.
  - For each weight a small adjustment is made by adding a small value, called a **delta value**, to the current weight, $\mathbf{w}[j]$
  - This adjustment should ensure that the change in the weight leads to a move *downward* on the error surface.
  - The **learning rate**, α, determines the size of the adjustments made to weights at each iteration of the algorithm
- The algorithm will converge to a point on the error surface where any subsequent changes to weights do not lead to a noticeably better model (within some tolerance).

Error Delta Function

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \mathbf{errorDelta}(\mathcal{D}, \mathbf{w}[j])$$

$$\text{, where } \mathbf{errorDelta}(\mathcal{D}, \mathbf{w}[j]) = \sum_{i=1}^{n} \left(\left(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \right) \times \mathbf{d}_i[j] \right)$$

- The **error delta function** calculates the **delta value** that determines the direction (either positive or negative) and the magnitude of the adjustments made to each weight.
- The direction and magnitude of the adjustment to be made to a weight is determined **by the gradient of the error surface** at the current position in the weight space.
- The gradient at any point on this error surface is given **by the value of the *partial derivative* of the error function** with respect to a particular weight at that point.

Error Delta Function Equation

$$\mathbf{errorDelta}(\mathcal{D}, \mathbf{w}[j]) = \sum_{i=1}^{n}\left(\left(t_i - \mathbb{M}_w(\mathbf{d}_i)\right) \times \mathbf{d}_i[j]\right)$$

- For simplicity, assume that $\mathcal{D}$ contains just <u>one training instance (**d**, *t*)</u> where **d** is a set of descriptive features and *t* is a target feature.
- The gradient of the error surface is given as the partial derivative of L_2 (MSE or SSE) loss function with respect to each weight, **w**[j]:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}[j]} L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) &= \frac{\partial}{\partial \mathbf{w}[j]}\left(\frac{1}{2}(t - \mathbb{M}_{\mathbf{w}}(\mathbf{d}))^2\right) \\
&= (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \times \frac{\partial}{\partial \mathbf{w}[j]}(t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \\
&= (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \times \frac{\partial}{\partial \mathbf{w}[j]}(t - (\mathbf{w} \cdot \mathbf{d})) \\
&= (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \times -\mathbf{d}[j]
\end{aligned}$$

By applying the differentiation **chain rule**

$\frac{\partial}{\partial \mathrm{w}[j]}(t - (\mathbf{w} \cdot \mathbf{d}))$ becomes $-\mathbf{d}[j]$. For example,

$$\left[\begin{aligned}
\mathbf{w} \cdot \mathbf{d} &= \mathbf{w}[0] \times \mathbf{d}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \mathbf{w}[2] \times \mathbf{d}[2] \\
&\quad + \mathbf{w}[3] \times \mathbf{d}[3] + \mathbf{w}[4] \times \mathbf{d}[4] \\
\frac{\partial}{\partial \mathbf{w}[0]} \mathbf{w} \cdot \mathbf{d} &= \frac{\partial}{\partial \mathbf{w}[0]}(\mathbf{w}[0] \times \mathbf{d}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \mathbf{w}[2] \times \mathbf{d}[2] \\
&\quad + \mathbf{w}[3] \times \mathbf{d}[3] + \mathbf{w}[4] \times \mathbf{d}[4]) \\
&= \mathbf{d}[0] + 0 + 0 + 0 + 0 = \mathbf{d}[0] \\
\frac{\partial}{\partial \mathbf{w}[4]} \mathbf{w} \cdot \mathbf{d} &= \frac{\partial}{\partial \mathbf{w}[4]}(\mathbf{w}[0] \times \mathbf{d}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \mathbf{w}[2] \times \mathbf{d}[2] \\
&\quad + \mathbf{w}[3] \times \mathbf{d}[3] + \mathbf{w}[4] \times \mathbf{d}[4]) \\
&= 0 + 0 + 0 + 0 + \mathbf{d}[4] = \mathbf{d}[4]
\end{aligned}\right.$$

, where **d**[*j*] is the *j*th descriptive feature of training instance (**d**, *t*)

Weight Update Rule

- When we ensure that the model moves towards minimizing the error by adjusting the weights in the opposite direction of the error gradient, and take into account multiple training instances, $(\mathbf{d}_1, t_1), (\mathbf{d}_2, t_2), \ldots (\mathbf{d}_n, t_n)$:

$$\frac{\partial}{\partial \mathbf{w}[j]} L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \sum_{i=1}^{n} \left(\left(t_i - \mathbb{M}_w(\mathbf{d}_i) \right) \times \mathbf{d}_i[j] \right)$$

$$\mathbf{errorDelta}(\mathcal{D}, \mathbf{w}[j]) = \sum_{i=1}^{n} \left(\left(t_i - \mathbb{M}_w(\mathbf{d}_i) \right) \times \mathbf{d}_i[j] \right)$$

- So, in each iteration, the weight associated with the jth descriptive feature is updated with

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \mathbf{errorDelta}(\mathcal{D}, \mathbf{w}[j])$$

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \sum_{i=1}^{n} \left(\left(t_i - \mathbb{M}_w(\mathbf{d}_i) \right) \times \mathbf{d}_i[j] \right)$$

, where $\mathbf{d}[j]$ is the jth descriptive feature of ith training instance $(\mathbf{d}_i, t_i)$

Weight Update Rule

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \underbrace{\sum_{i=1}^{n}\left(\left(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)\right) \times \mathbf{d}_i[j]\right)}_{errorDelta(\mathcal{D},\, \mathbf{w}[j])}$$

- The weight update rule adjusts the weights of the model based on the discrepancies between the predictions and the actual outcomes.
  - When predictions are generally too high:
    - Decrease $\mathbf{w}[j]$ if $\mathbf{d}_i[j]$ is positive.
    - Increase $\mathbf{w}[j]$ if $\mathbf{d}_i[j]$ is negative.
  - When predictions are generally too low:
    - Increase $\mathbf{w}[j]$ if $\mathbf{d}_i[j]$ is positive.
    - Decrease $\mathbf{w}[j]$ if $\mathbf{d}_i[j]$ is negative.
- This approach ensures that the model fine-tunes itself to get closer to the correct predictions by adjusting the weights in the right direction.

Characteristics of Gradient Descent

- Gradient descent is widely utilized for its simplicity and reasonable efficiency in training multivariate linear regression models.

- **Inherent Inductive Biases**:
  - **Preference Bias**: The algorithm inherently prefers models that minimize the loss function, the Sum of Squared Errors (SSE).
  - **Restriction Bias**:
    - Considers only linear combinations of descriptive features.
    - Pursues a singular path through the error gradient, originating from a random starting point.

Outline

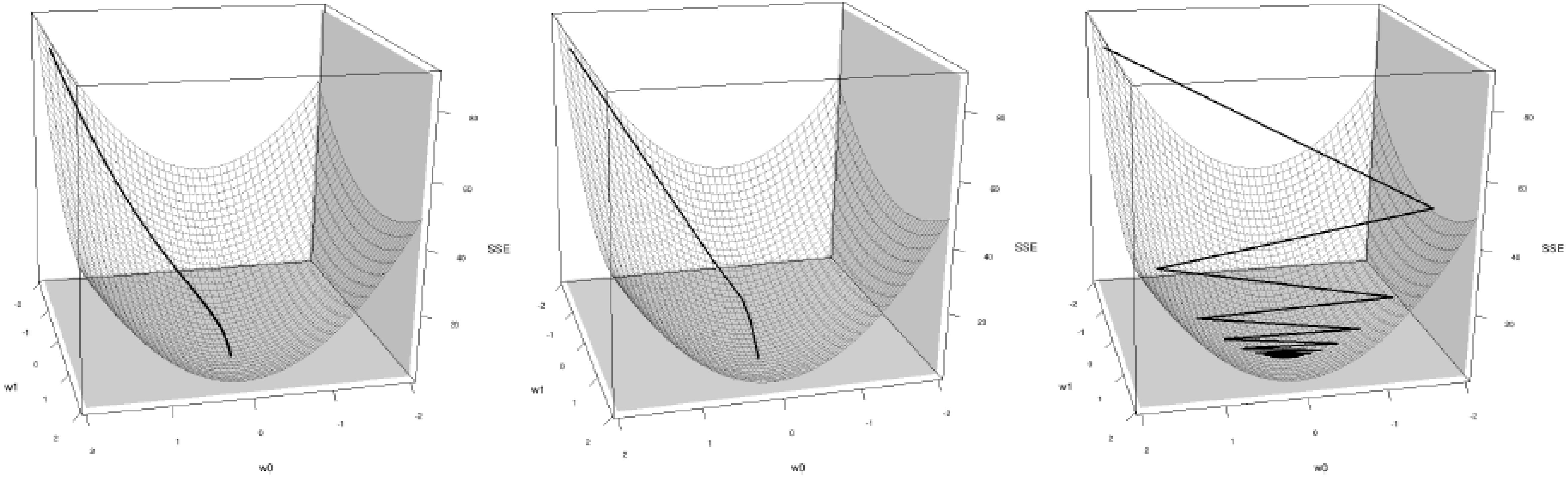
- Fundamentals
- Standard Approach: Multivariate Linear Regression with Gradient Descent
  - Multivariate Linear Regression
  - Gradient Descent Algorithm
  - ☞ **Choosing Learning Rates & Initial Weights**
  - A Worked Example
- Extensions and Variations

Choosing Learning Rates & Initial Weights

- **Selecting learning rates.**
  - The learning rate, α, governs the magnitude of weight adjustments at every step in the optimization process.
  - The optimal choice of learning rates lacks a strict definition.
  - Commonly, practitioners rely on experiential guidelines.
  - Learning rates usually fall within the range [0.00001, 10]

- **Selecting initial weights.**
  - Empirical studies suggest that selecting random initial weights evenly from the range [-0.2, 0.2] generally yields satisfactory results.

Example



(a) (b) (c)

Figure: Plots of the journeys made across the error surface for the office rentals prediction problem for different learning rates: (a) a very small learning rate (0.002), (b) a medium learning rate (0.08) and (c) a very large learning rate (0.18).

Outline

- Fundamentals
- Standard Approach: Multivariate Linear Regression with Gradient Descent
  - Multivariate Linear Regression
  - Gradient Descent Algorithm
  - Choosing Learning Rates & Initial Weights
  - ☞ **A Worked Example**
- Extensions and Variations

A Worked Example

- Dataset

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

Table. A dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-center offices.

- Linear regression equation for rental price prediction

$$\text{RENTAL PRICE} = \mathbf{w}[0] + \mathbf{w}[1] \times \text{SIZE} + \mathbf{w}[2] \times \text{FLOOR} + \mathbf{w}[3] \times \text{BROADBAND RATE}$$

Initial Weight

- Let the learning rate $\alpha = 0.00000002$
- The initial weights are chosen from a uniform random distribution in [-0.2, 0.2]
  - Suppose the initial weights

Initial Weights

w[0]:	-0.146	**w**[1]:	0.185	**w**[2]:	-0.044	**w**[3]:	0.119

First Iteration

- Current weights:

Weights

w[0]:	-0.146	**w**[1]:	0.185	**w**[2]:	-0.044	**w**[3]:	0.119

- Make predictions using the initial model,
- The weights are updated: $$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^{n} \left((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j] \right)}_{errorDelta(\mathcal{D},\mathbf{w}[j])}$$
  - Calculate the delta value for each weight, errorDelta($\mathcal{D}$, w[j]) and then update the weight, e.g., **w**[1]← 0.185 + 0.00000002 × 2,412,073.90 = 0.23324148
- New weights:

New Weights (Iteration 1)

w[0]:	-0.146	**w**[1]:	0.233
w[2]:	-0.043	**w**[3]:	0.121

Iteration 1

ID	Rental Price	Pred.	Error	Squared Error	errorDelta($\mathcal{D}$, **w**[i]) **w**[0]	**w**[1]	**w**[2]	**w**[3]
1	320	93.26	226.74	51411.08	226.74	113370.05	906.96	1813.92
2	380	107.41	272.59	74307.70	272.59	149926.92	1908.16	13629.72
3	400	115.15	284.85	81138.96	284.85	176606.39	2563.64	1993.94
4	390	119.21	270.79	73327.67	270.79	170598.22	1353.95	6498.98
5	385	134.64	250.36	62682.22	250.36	166492.17	2002.91	25036.42
6	410	130.31	279.69	78226.32	279.69	195782.78	1118.76	2237.52
7	480	142.89	337.11	113639.88	337.11	259570.96	3371.05	2359.74
8	600	168.32	431.68	186348.45	431.68	379879.24	5180.17	21584.05
9	570	170.63	399.37	159499.37	399.37	367423.83	5591.23	3194.99
10	620	187.58	432.42	186989.95	432.42	432423.35	3891.81	10378.16
			Sum	1067571.59	3185.61	2412073.90	27888.65	88727.43
Sum of squared errors (Sum/2)				533785.80				

Second Iteration

- Current weights:

Weights							
w[0]:	-0.146	**w**[1]:	0.233	**w**[2]:	-0.043	**w**[3]:	0.121

- Make predictions using current weights,
- Weight update: $\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^{n} \left(\left(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \right) \times d_i[j] \right)}_{errorDelta(\mathcal{D}, \mathbf{w}[j])}$
    - e.g., **w**[1]← 0.233 + 0.00000002 × 2195615.84= 0.27691232

- New weights:

New Weights (Iteration 2)			
w[0]:	-0.145	**w**[1]:	0.277
w[2]:	-0.043	**w**[3]:	0.123

Iteration 2

ID	Rental Price	Pred.	Error	Squared Error	errorDelta($\mathcal{D}$, **w**[i]) **w**[0]	**w**[1]	**w**[2]	**w**[3]
1	320	117.40	202.60	41047.92	202.60	101301.44	810.41	1620.82
2	380	134.03	245.97	60500.69	245.97	135282.89	1721.78	12298.44
3	400	145.08	254.92	64985.12	254.92	158051.51	2294.30	1784.45
4	390	149.65	240.35	57769.68	240.35	151422.55	1201.77	5768.48
5	385	166.90	218.10	47568.31	218.10	145037.57	1744.81	21810.16
6	410	164.10	245.90	60468.86	245.90	172132.91	983.62	1967.23
7	480	180.06	299.94	89964.69	299.94	230954.68	2999.41	2099.59
8	600	210.87	389.13	151424.47	389.13	342437.01	4669.60	19456.65
9	570	215.03	354.97	126003.34	354.97	326571.94	4969.57	2839.76
10	620	187.58	432.42	186989.95	432.42	432423.35	3891.81	10378.16
			Sum	886723.04	2884.32	2195615.84	25287.08	80023.74
Sum of squared errors (Sum/2)				443361.52				

Final Weights

- The algorithm then keeps iteratively applying the weight update rule **until it converges on a stable set of weights** beyond which little improvement in model accuracy is possible.
- After 100 iterations the final values for the weights are:

w[0] = -0.1513,

w[1] = 0.6270,

w[2] = -0.1781

w[3] = 0.0714

which results in a sum of squared errors value of 2,913.5

Outline

- Fundamentals
- Standard Approach: Multivariate Linear Regression with Gradient Descent
- ☞ **Extensions and Variations**
  - Interpreting Linear Regression Models
  - Assessing the Parameter Estimates
  - Setting the Learning Rate Using Weight Decay
  - Handling Categorical Descriptive Features
  - Handling Categorical Target Features: Logistic Regression
  - Modeling Non-linear Relationships
  - Multinomial Logistic Regression

Interpreting Linear Regression Models

- The **weights** used by linear regression models **indicate the effect of each descriptive feature on the predictions** returned by the model.
  - The **signs** of the weights indicate whether different descriptive features have a positive or a negative impact on the prediction.
  - The **magnitudes** of the weights show how much the value of the target feature changes for a unit change in the value of a particular descriptive feature.
- The descriptive features associated with higher weights are more predictive than those with lower weights. However, <u>direct comparison of the weights can be a mistake,</u> <u>especially when the descriptive features have varying scale.</u>

Accuracy on the Parameter Estimates

- In the simple regression model, $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$, **how close $\widehat{\boldsymbol{\beta}}_0$ and $\widehat{\boldsymbol{\beta}}_1$ are to the true values $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_1$?**
- We compute the ***standard error* (*SE*)** **of an estimator** which reflects how it varies under repeated sampling.
- The ***standard errors SE*** **associated with $\widehat{\boldsymbol{\beta}}_0$ and $\widehat{\boldsymbol{\beta}}_1$** are computed as:

$$Var(\hat{\beta}_1) = \boldsymbol{SE(\widehat{\beta}_1)^2} = \frac{\boldsymbol{\sigma^2}}{\sum_{i=1}^{n}(x_i-\bar{x})^2} = \frac{\boldsymbol{RSE^2}}{\sum_{i=1}^{n}(x_i-\bar{x})^2} \quad , \boldsymbol{SE(\widehat{\beta}_1)} = \frac{\boldsymbol{RSE}}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}}$$

$$Var(\hat{\beta}_0) = \boldsymbol{SE(\widehat{\beta}_0)^2} = \boldsymbol{\sigma^2}\left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}\right],$$ where $\boldsymbol{\sigma^2 = Var(\epsilon)}$.

  - $\boldsymbol{\sigma}$ **can be estimated with *residual standard error* of the model,** $RSE = \sqrt{RSS/(n-2)}$,

    where $RSS = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.

    So $\boldsymbol{\sigma = RSE = \sqrt{RSS/(n-2)} = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n-2}}}$

  - In the case of the stand error associate $\hat{\beta}_1$, $SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2} = \frac{RSE^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$. So, $SE(\hat{\beta}_1) = \frac{RSE}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}}$

Confidence Intervals of True Parameter Values

- ***Standard Errors*** (SE) can also be used to compute ***confidence intervals*, a range which will contain the true unknown value of the parameters**
- For linear regression, **the 95% confidence interval for $\boldsymbol{\beta_1}$**

$$[\hat{\beta}_1 - 2 \cdot SE(\hat{\beta}_1),\ \hat{\beta}_1 + 2 \cdot SE(\hat{\beta}_1)]$$

will contain the true value of β_1 with approximately a 95% chance

- Similarly, the 95% confidence interval for β_0

$$[\hat{\beta}_0 - 2 \cdot SE(\hat{\beta}_0),\ \hat{\beta}_0 + 2 \cdot SE(\hat{\beta}_0)]$$

Statistical Significance Test

- A **statistical significance test** can be used **to determine the importance of each descriptive feature in the model**
- A statistical significance test works by stating a *null hypothesis* and then determining whether there is enough evidence to accept or reject this hypothesis.

General Procedure of Hypothesis Test

1. State a **null hypothesis**
2. A **test-statistic** is computed
3. The ***p*-value** is computed.
   - A *p*-value is a statistical measurement used **to validate a hypothesis against observed data.**
   - A *p*-value measures **the probability of obtaining the observed results, assuming that the null hypothesis is true.**
4. **The *p-value* is compared to a predefined significant threshold** (typically 5% or 1%)
   - A *p*-value of 0.05 or lower is generally considered statistically significant.
5. If the ***p-value*** is **less than or equal to the threshold**, the **null hypothesis is rejected.**
   - **A smaller p-value** means that there is **stronger evidence in favor of the alternative hypothesis**.
   - A *p*-value can serve as an alternative to or in addition to preselected confidence levels for hypothesis testing.

Hypothesis Testing for Simple Linear Regression

For the simple linear regression model, $Y \approx \beta_0 + \beta_1 X$

- The most common hypothesis test involves testing the ***null hypothesis*** of

$$\boldsymbol{H_0}\text{: There is no relationship between X and Y}$$

vs. the ***alternative hypothesis***

$$\boldsymbol{H_A}\text{ : There is some relationship between X and Y}$$

- This corresponds to testing

$$\boldsymbol{H_0}: \boldsymbol{\beta_1} = \boldsymbol{0} \quad \text{vs.} \quad \boldsymbol{H_A}: \boldsymbol{\beta_1} \neq \boldsymbol{0}$$

since if $\beta_1 = 0$ then the model reduces to $Y = \beta_0$, and X is not associated with Y.

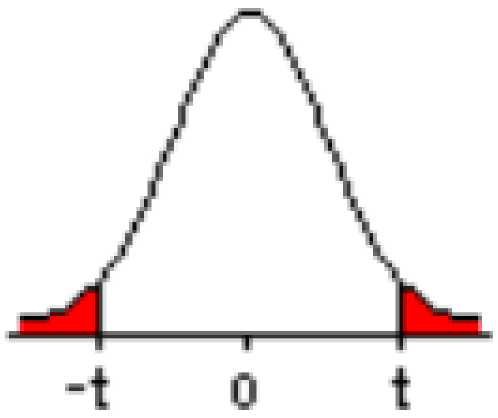
Hypothesis Testing

- **To perform hypothesis tests on the parameters**, we need to **determine whether $\hat{\beta}_1$ is sufficiently far from zero. How far is far enough?**
  - This depends on the accuracy of $\hat{\beta}_1$, i.e., the standard error of $\hat{\beta}_1$, $SE\left(\hat{\beta}_1\right)$
- We calculate a ***t*-statistic** that measures how many standard deviations $\hat{\beta}_1$ is from 0.

$$t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)} = \frac{\hat{\beta}_1}{SE(\hat{\beta}_1)}$$

- $SE(\hat{\beta}_1) = \frac{RSE}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}} = \sqrt{\frac{1}{n-2} \times \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$, where $RSE = \sqrt{RSS/(n-2)} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}}$
- **If there really is no relationship between *X* and *Y*,** then we expect that **the *t*-statistic will have a *t*-distribution with *n* – 2 degrees of freedom (*df*)**

Hypothesis Testing (Cont.)



- The ***t*-distribution** has a bell shape and for values of n greater than approximately 30, it is quite similar to the normal distribution.
- For the hypothesis tests, find the ***p-value*** which is the probability of observing any number equal to or larger than $|t|$, assuming there is no relationship between X and Y ($\beta_1 = 0$).
  - Using a standard t-statistic look-up table, we can then determine the p-value associated with this test (this is a two tailed t-test with degrees of freedom n - 2).
- If the *p-value* is small, we can infer that there is an association between X and Y
  - **If the *p-value* is less than 5%,** we ***reject the null hypothesis*** ($H_0 : \beta_1 = 0$) – that is, we declare a relationship to exist between X and Y
    - With a P value of 5%, there is only a 5% chance that the result from a random distribution, and a 95% chance that there is an association between X and Y.
  - Typical *p-value* cutoffs for rejecting null hypothesis are 5% or 1%
  - When df (= n - 2) = 30, these correspond to t-statistics of around 2 and 2.75, respectively.

Statistical Significance Test for Multivariable Linear Regression

- The statistical significance test we use to **analyze the importance of a descriptive feature d [*j*] in a multivariable linear regression model** is ***t*-test**.

- The test hypotheses using our weight notation are:

$$\boldsymbol{H_0}: \mathbf{w}[j] = \mathbf{0} \quad \text{vs.} \quad \boldsymbol{H_A}: \mathbf{w}[j] \neq \mathbf{0}$$

- Compute a ***t*-statistic**, $\boldsymbol{t} = \dfrac{\mathbf{w}[j]}{\boldsymbol{SE}(\mathbf{w}[j])}$, where $SE(\mathrm{w}[j]) = \sqrt{\dfrac{1}{n-2} \times \dfrac{\sum_{i=1}^{n}(y_i - \mathbb{M}_w(\mathrm{d}[\mathrm{j}]))^2}{\sum_{i=1}^{n}(\mathbf{d}_i[j] - \bar{\mathbf{d}}[j])^2}}$

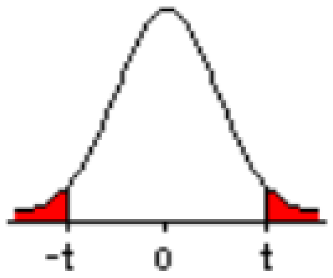
- Determine the ***p*-value** associated with this *t*-statistic from a *t*-distribution with *n* – 2 degrees of freedom (*df*)

- If the *p*-value is less than the required significance level, typically 0.05 (5%), we reject the null hypothesis and say that the descriptive feature has a significant impact on the model.

Example

- The linear regression model (*least squares model*) for the regression of size, floor and broadband rate on office rental price

Descriptive Feature	Weight	Standard Error	t-statistic	p-value
SIZE	0.6270	0.0545	11.504	$<$0.0001
FLOOR	-0.1781	2.7042	-0.066	0.949
BROADBAND RATE	0.071396	0.2969	0.240	0.816



- The weight of SIZE, **w**[1], is very large relative to its standard error, so the ***t*-statistic,** $t = \frac{\mathbf{w}[1]}{SE(\mathbf{w}[1])}$, **is also large** ; ***p*-value will be very small.**
- Hence we can **reject** $\boldsymbol{H_0}$ and **conclude that** $\mathbf{w}[1] \neq \mathbf{0}$, i.e., Office size has an effect.
  - The probability (t-statistic) of seeing such values if H_0 is true (i.e., no effect) are virtually zero, i.e., 0.0001

Statistical Significance Test with All Descriptive Features

- Test Hypothesis

$$H_0: \mathbf{w}[1] = \mathbf{w}[2] \ldots \mathbf{w}[m] = \mathbf{0} \quad \text{vs.} \quad H_A: \text{at least one } \mathbf{w}[j] \neq \mathbf{0}$$

- The hypothesis test is performed by computing the ***F*-statistic**,

$$F = \frac{(TSS - RSS)/d}{RSS/(n-d-1)} \sim F_{d,n-d-1}$$

, where $TSS = \sum(t_i - \bar{t})^2$, $RSS = \sum(t_i - \mathbb{M}_w(\boldsymbol{d}_i))^2$

- **If the F-statistic value is close to 1**, we can conclude there is **no relationship** between the target feature and descriptive features

- **Example**:

Quantity	Value
Residual standard error	1.69
R^2	0.897
F-statistic	570

Outline

- Fundamentals
- Standard Approach: Multivariate Linear Regression with Gradient Descent
- Extensions and Variations
  - Interpreting Multivariable Linear Regression Models
  - Assessing the Parameter Estimates
  - ☞ **Setting the Learning Rate Using Weight Decay**
  - Handling Categorical Descriptive Features
  - Handling Categorical Target Features: Logistic Regression
  - Modeling Non-linear Relationships
  - Multinomial Logistic Regression

Learning Rate Decay

- In the gradient decent algorithm, the **learning rate** (denoted by α) determines the size of the adjustment applied to each weight in every iteration of process, aiming to find the optimal weight set.
- Typically, a fixed learning rate is selected from [0.00001, 10], e.g., 0.1, 0.01, or 0.001
- A more systematic approach is to use **learning rate decay** which allows the learning rate to start at a large value and then decay over time according to a predefined schedule.
- A good approach is to use the following decay schedule: $\boldsymbol{\alpha_\tau = \alpha_0 \frac{c}{c+\tau}}$
  - Where α_0 is an initial learning rate (e.g., 1.0), ***c*** is a constant that controls how quickly the learning rate decays (the value of this parameter depends on how quickly the algorithm converges, but it is often set to quite a large value, e.g., 100), and τ is the current iteration of the gradient descent algorithm.
- **Learning rate decay almost always leads to better performance than a fixed learning rate.** However it still requires that problem-dependent values are chosen for α_0 and c

Example without/with Learning Rate Decay

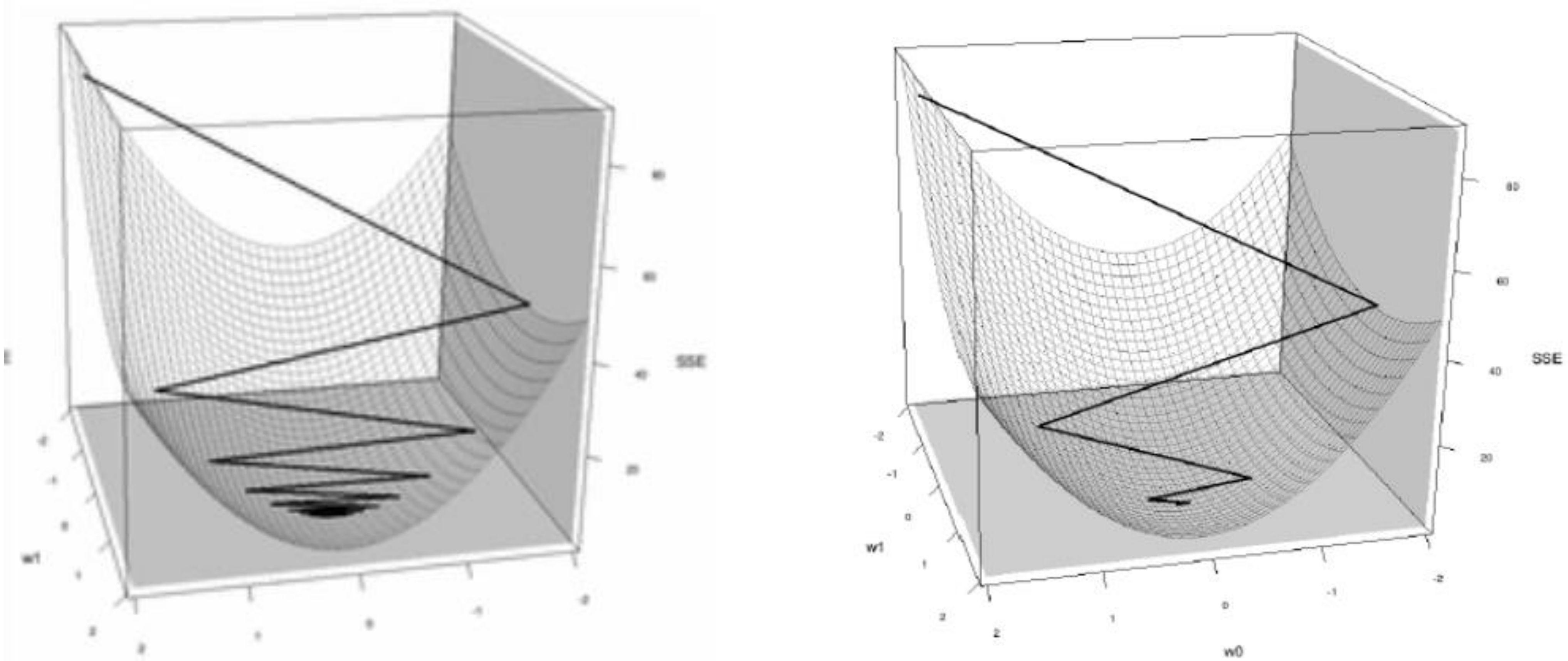


Figure: (a) The journey made across the error surface for the simple office rentals prediction problem for **a fixed learning rate** (α =0.18). (b) The journey across the error surface when **learning rate decay** is used (σ_0= 0.18, c = 10)

Example with Learning Rate Decay

- **Utilizing learning rate decay can also mitigate issues related to excessively large error rates,** which may cause the sum of squared errors to increase instead of decrease.

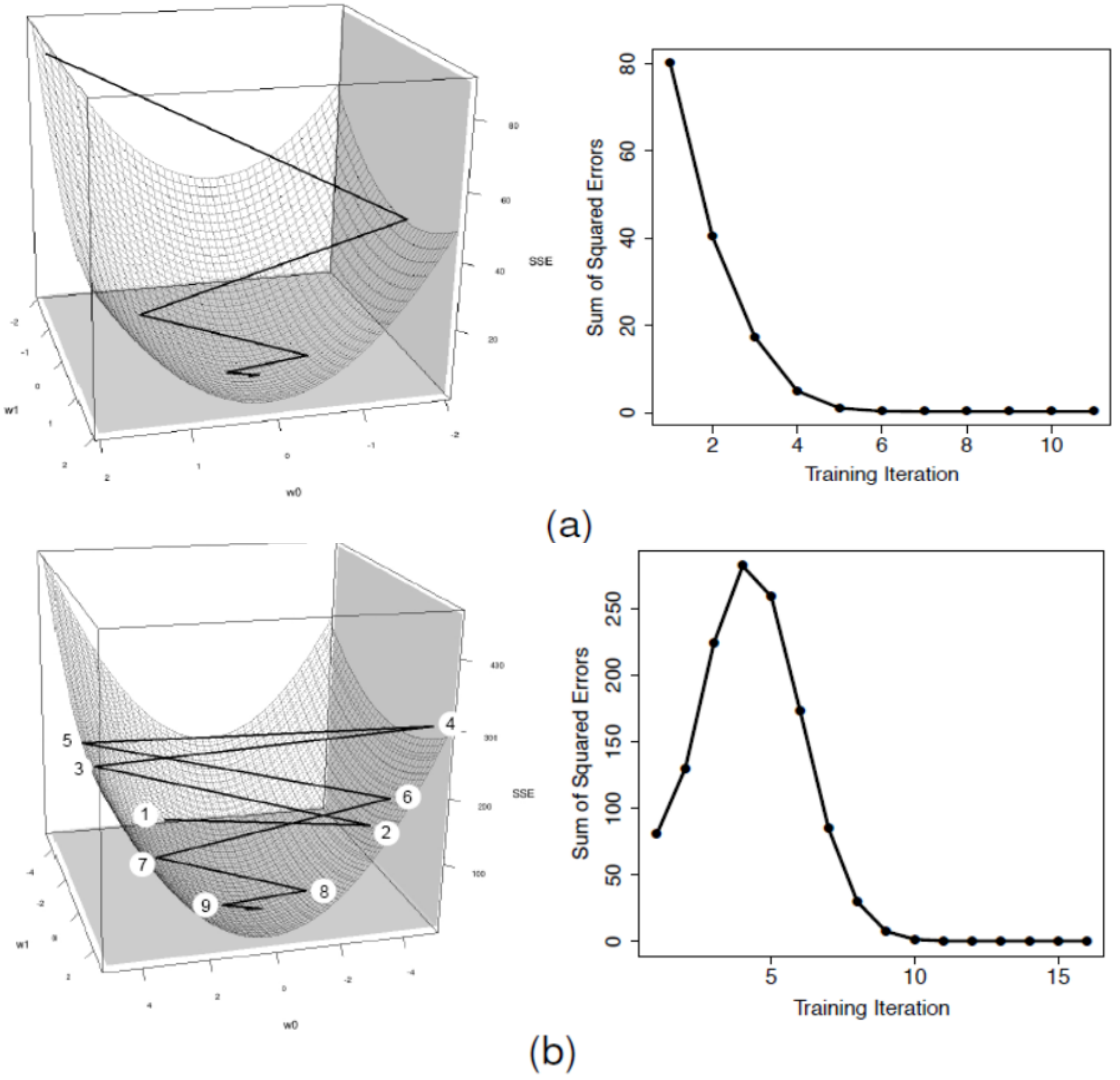


Figure: (a) The journey across the error surface for the office rentals prediction problem when learning rate decay is used **(α_0= 0.18, c = 10)** and a plot of the changing sum of squared error values during this journey. (b) The journey across the error surface for the office rentals prediction problem when learning rate decay is used **(α_0= 0.25, c = 100)** and a plot of the changing sum of squared error values during this journey.

Outline

- Fundamentals
- Standard Approach: Multivariate Linear Regression with Gradient Descent
- Extensions and Variations
  - Interpreting Multivariable Linear Regression Models
  - Assessing the Parameter Estimates
  - Setting the Learning Rate Using Weight Decay
  - ☞ **Handling Categorical Descriptive Features**
  - Handling Categorical Target Features: Logistic Regression
  - Modeling Non-linear Relationships
  - Multinomial Logistic Regression

Categorical Features

- The basic structure of the multivariable linear regression model allows for only continuous descriptive features
- **The most common approach to handling categorical features** uses a transformation that converts a single categorical descriptive feature into a number of continuous descriptive feature values that can encode the levels of the categorical feature.

Example

- The ENERGY RATING descriptive feature would be converted into **three new continuous features**, as it has 3 distinct levels: ’A’, ’B’, or ’C’.
- The regression equation for this RENTAL PRICE model would change to

$$
\begin{aligned}
\text{RENTAL PRICE} = \mathbf{w}[0] &+ \mathbf{w}[1] \times \text{SIZE} + \mathbf{w}[2] \times \text{FLOOR} \\
&+ \mathbf{w}[3] \times \text{BROADBAND RATE} \\
&+ \mathbf{w}[4] \times \text{ENERGY RATING A} \\
&+ \mathbf{w}[5] \times \text{ENERGY RATING B} \\
&+ \mathbf{w}[6] \times \text{ENERGY RATING C}
\end{aligned}
$$

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

ID	Size	Floor	Broadband Rate	Energy Rating A	Energy Rating B	Energy Rating C	Rental Price
1	500	4	8	0	0	1	320
2	550	7	50	1	0	0	380
3	620	9	7	1	0	0	400
4	630	5	24	0	1	0	390
5	665	8	100	0	0	1	385
6	700	4	8	0	1	0	410
7	770	10	7	0	1	0	480
8	880	12	50	1	0	0	600
9	920	14	8	0	0	1	570
10	1 000	9	24	0	1	0	620

Table: The office rentals dataset adjusted to handle the categorical ENERGY RATING descriptive feature in linear regression models.

Outline

- Fundamentals
- Standard Approach: Multivariate Linear Regression with Gradient Descent
- Extensions and Variations
  - Interpreting Multivariable Linear Regression Models
  - Assessing the Parameter Estimates
  - Setting the Learning Rate Using Weight Decay
  - Handling Categorical Descriptive Features
  - ☞ **Handling Categorical Target Features: Logistic Regression**
  - Modeling Non-linear Relationships
  - Multinomial Logistic Regression

Prediction Problem with Categorical Target Features

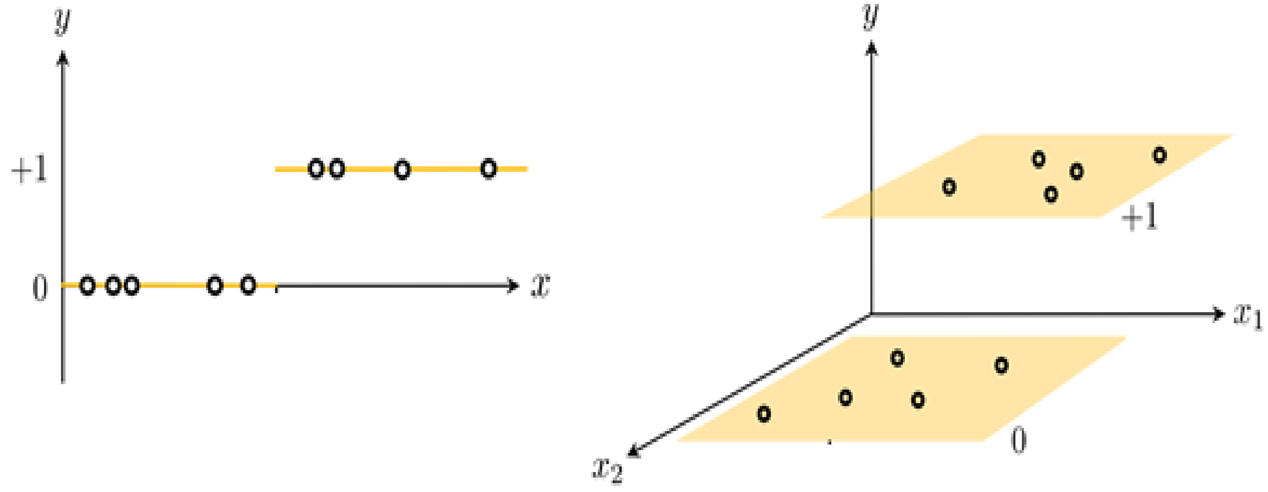
- The linear regression model assumes that the target feature is *continuous (quantitative)*.
- But in many situations, the target feature is *categorical* (*qualitative*) (which takes category (class) values in an unordered set, such as email $\in$ {spam; ham}.
- **How to adjust the multivariable linear regression to handle categorical target features?**

Regression with Categorical Target Features

- The data is in the form of n input/output pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where each input $\mathbf{x}_i$ is an d-dimensional vector, and the corresponding $\boldsymbol{y_i}$ **takes only two discrete numbers, e.g.,** $\boldsymbol{y_i \in [0, +1]}$

- The problem of two-class classification can be viewed as a case of ***nonlinear* regression.**

- **The goal is to regress (or fit) a *nonlinear* step function to the data.**



(a) one-dimensional input **x** (b) two-dimensional input **x**

Example Dataset

ID	RPM	VIBRATION	STATUS	ID	RPM	VIBRATION	STATUS
1	568	585	good	29	562	309	faulty
2	586	565	good	30	578	346	faulty
3	609	536	good	31	593	357	faulty
4	616	492	good	32	626	341	faulty
5	632	465	good	33	635	252	faulty
6	652	528	good	34	658	235	faulty
7	655	496	good	35	663	299	faulty
8	660	471	good	36	677	223	faulty
9	688	408	good	37	685	303	faulty
10	696	399	good	38	698	197	faulty
11	708	387	good	39	699	311	faulty
12	701	434	good	40	712	257	faulty
13	715	506	good	41	722	193	faulty
14	732	485	good	42	735	259	faulty
15	731	395	good	43	738	314	faulty
16	749	398	good	44	753	113	faulty
17	759	512	good	45	767	286	faulty
18	773	431	good	46	771	264	faulty
19	782	456	good	47	780	137	faulty
20	797	476	good	48	784	131	faulty
21	794	421	good	49	798	132	faulty
22	824	452	good	50	820	152	faulty
23	835	441	good	51	834	157	faulty
24	862	372	good	52	858	163	faulty
25	879	340	good	53	888	91	faulty
26	892	370	good	54	891	156	faulty
27	913	373	good	55	911	79	faulty
28	933	330	good	56	939	99	faulty

Table: A generators dataset with a categorical feature, STATUS, which indicates ‘good’ or ‘faculty’ the day after two measurements were taken.

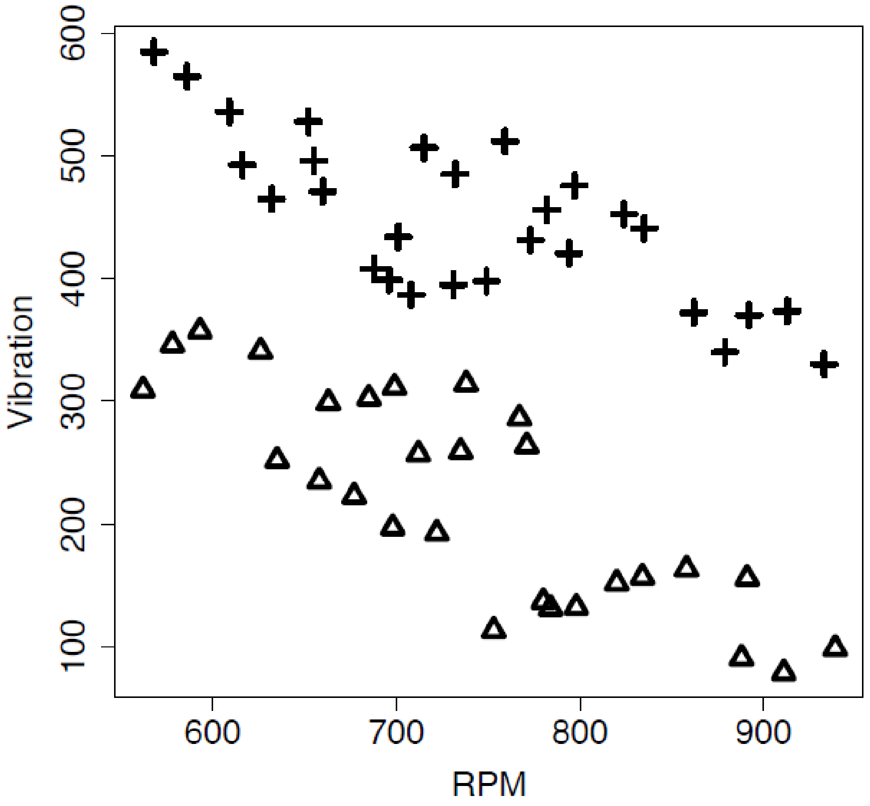


Figure: A scatter plot of the RPM and VIBRATION descriptive features from the generators dataset where **’good’** generators are shown as **crosses** and **’faulty’** generators are shown as **triangles**.

Decision Boundary – Linear Separator

- A **decision boundary** shows a separation between the two types of class instances.
  - E.g., The generator dataset is linearly separable in terms of the two descriptive features RPM and VIBRATION
- As the decision boundary is a **linear separator** it can be defined using the **equation of the line** as:

$$\text{VIBRATION} = 830 - 0.667 \times \text{RPM}$$

  - For the points on the decision boundary

$$830 - 0.667 \times \text{RPM} - \text{VIBRATION} = \mathbf{0}$$

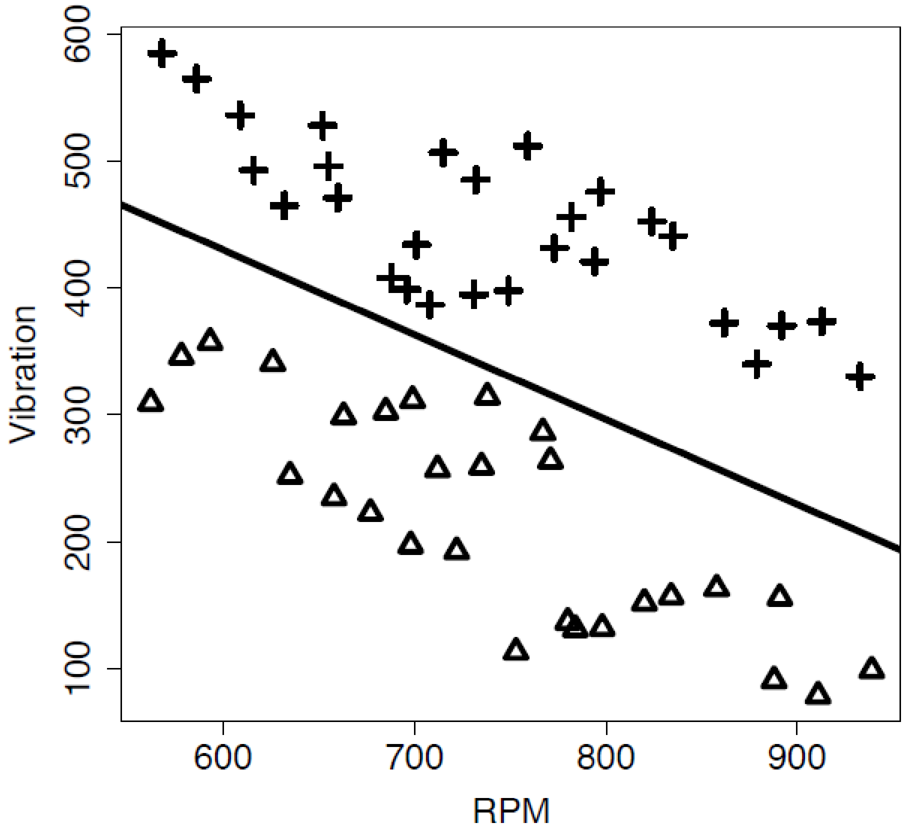


Figure: The scatter plot of the RPM and VIBRATION with a **decision boundary** separating ’good’ generators (crosses) from ’faulty’ generators (triangles) is also shown.

Decision Boundary and Categorical Target Prediction

- Moreover, this linear model well behaves with following

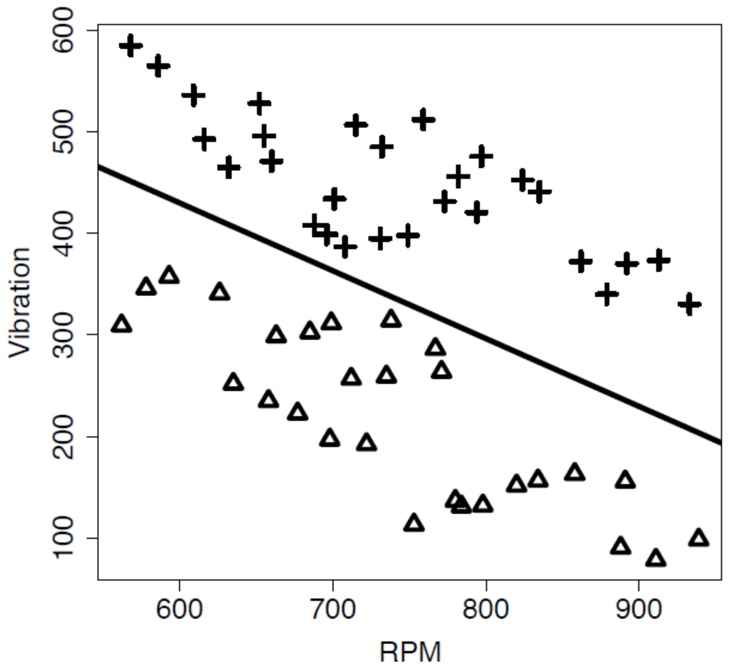
  VIBRATION = 830 – 0.667 × RPM

  - All the data points above the decision boundary will result in a **negative** value when plugged in to the decision boundary equation.
    - For the instance RPM = 810, VIBRATION = 495, the result is:

      830 – 0.667 × 801 - 495 = -205.27

  - All the data points below the decision boundary will result in a **postive** value when plugged in to the decision boundary equation.
    - For the instance RPM = 650 and VIBRATION = 240, the result is:

      830 – 0.667 × 650 - 240 = 156.45



- If the 'good' and 'faulty' target feature levels are represented as 0 and 1, we can use **the model to predict the categorical target feature**

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{d} \geq 0 \\ 0 & \textit{otherwise} \end{cases}$$

, where **d** is a set of descriptive features for an instance, **w** is the set of weights in the model

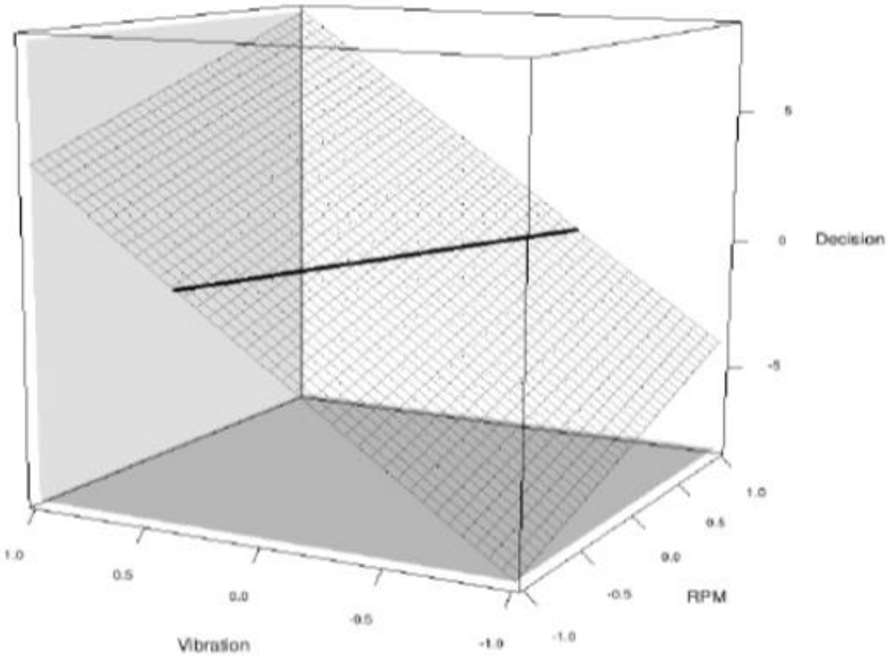
Decision Surface

- A model to predict a categorical target feature:

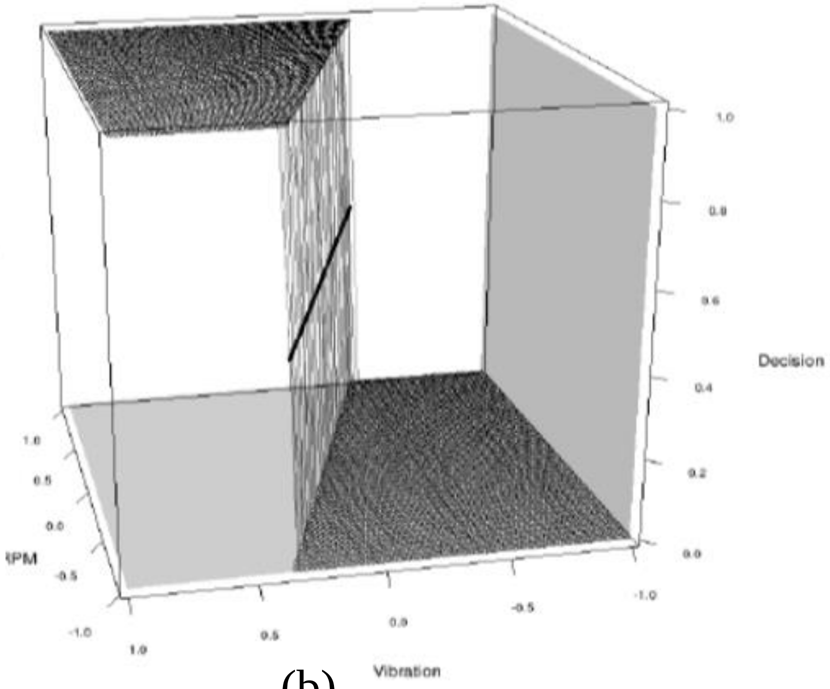
$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{d} \geq 0 \\ 0 & \textit{otherwise} \end{cases}$$

, where **d** is a set of descriptive features for an instance, **w** is the set of weights in the model

- The surface defined by this rule is known as a **decision surface**.



(a)

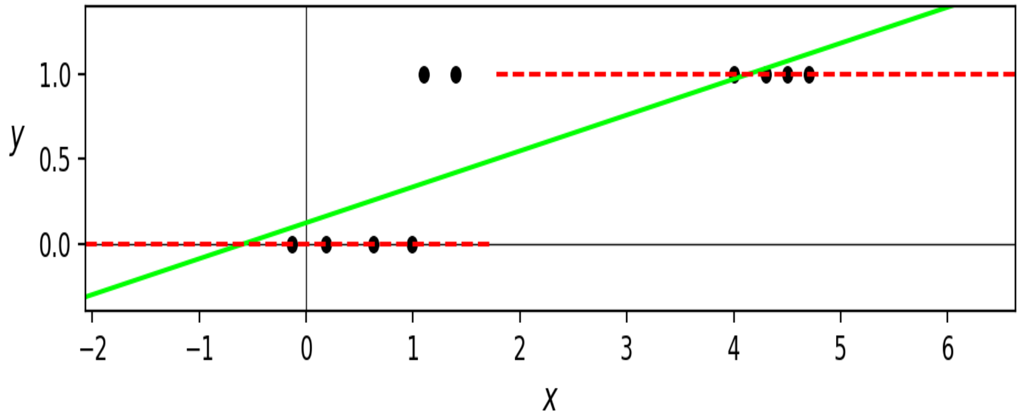


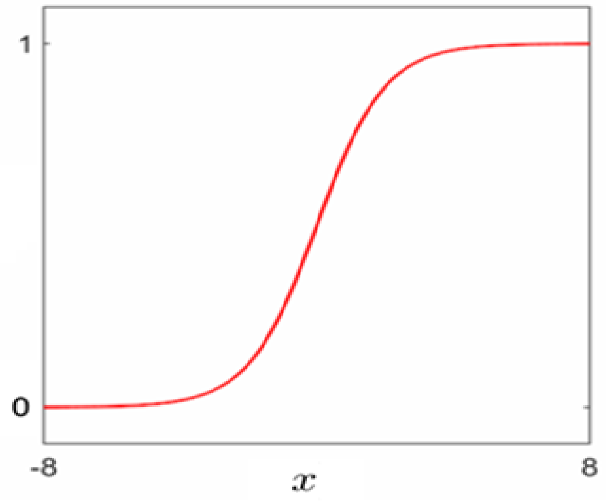
(b)

Figure: (a) **A surface showing the value (w · d) of model equation** for all values of RPM and VIBRATION. The decision boundary given in Equation is highlighted.
(b) **The same surface linearly thresholded at zero to operate as a predictor.** - the surface is used to make predictions by delineating where values are categorized either above or below zero

Fitting a Discontinuous Decision Boundary

- How to determine the values for the weights, **w**, that will minimize the error function of model $\mathbb{M}_w(\mathbf{d})$?
- **Issues with Discontinuity**
  - Minimizing the sum of squared errors is notably difficult due to the **discontinuous decision boundary**.
  - The **non-differentiable nature of the boundary** inhibits our ability to calculate the gradient of the error surface.
- The model produces highly confident predictions (either 0 or 1), where a measured, subtle approach might be more apt.





Logistic Function

- So, we need to **replace the step function with a continuous approximate** that matches closely.
- Often we are more interested in **estimating the *probabilities* that X belongs to each category in the target.**
  - With one predator X, output the form of probabilities of the occurrence of a target $\mathrm{p}(X) = \Pr(Y = 1|X)$.
  - $\mathrm{p}(X)$ must output between 0 and 1 for all values of X
- One of functions which meet this requirement is ***logistic function*** (***logistic sigmode function***) is

$$logistic(x) = \frac{1}{1+e^{-x}}$$

, where x is a numeric value and e is Euler's number ($\cong$ 2.7183)

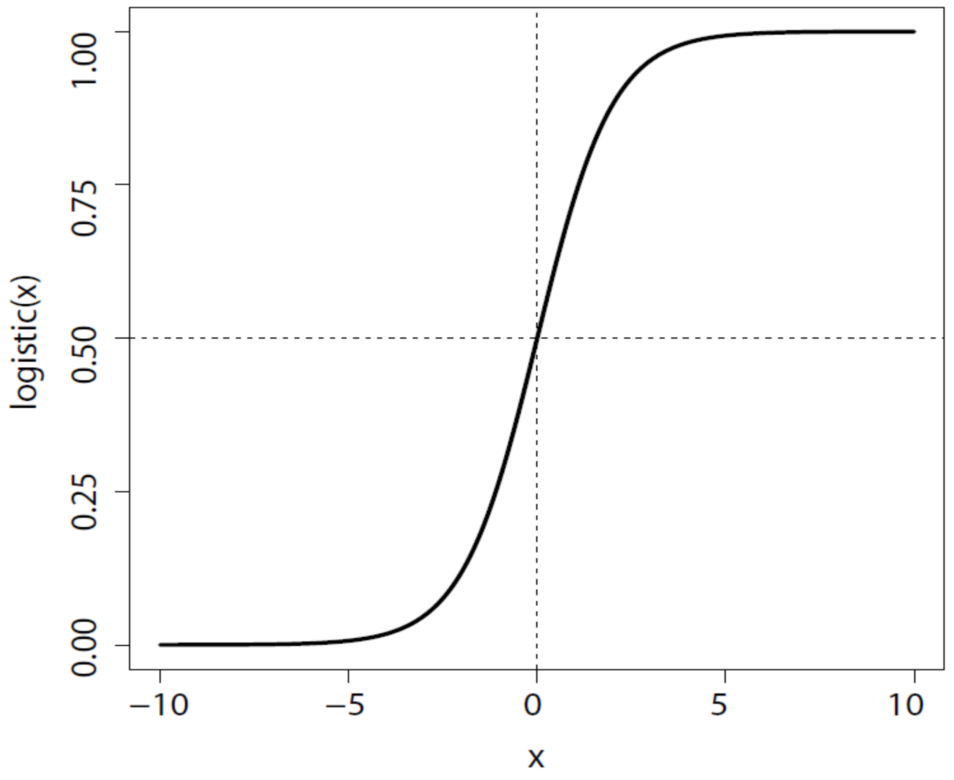
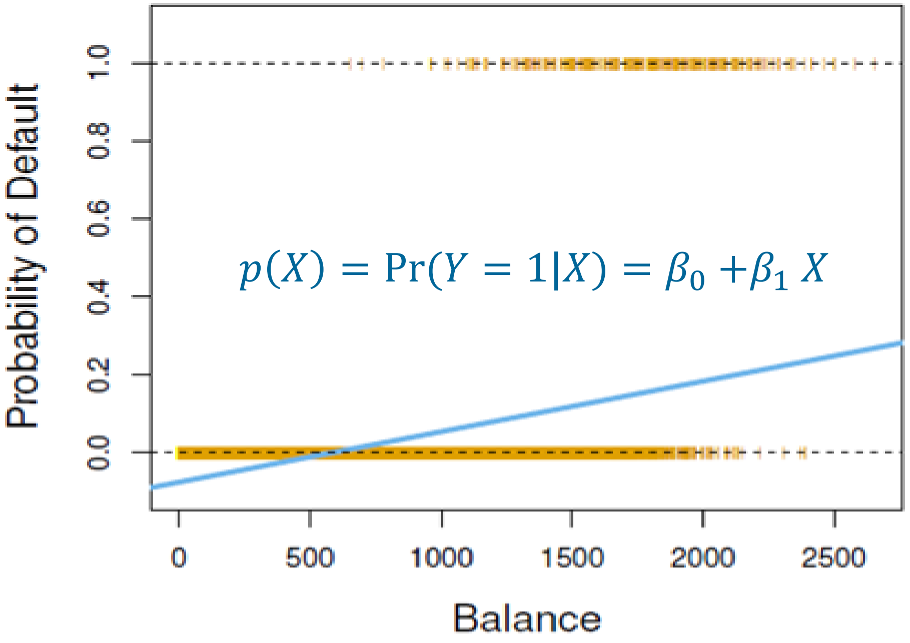
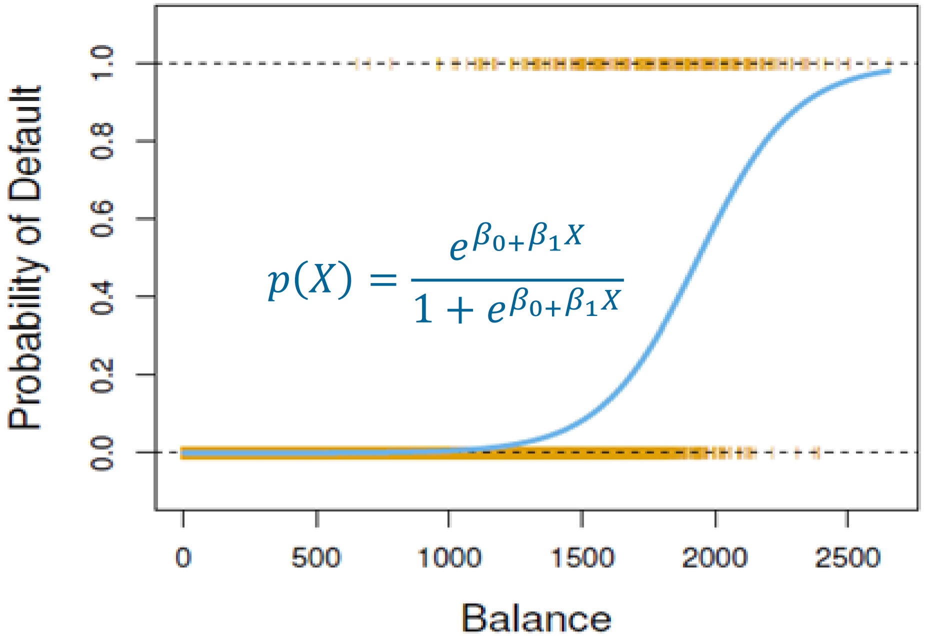


Figure. A plot of the logistic function for the range of values [-10, 10]

Linear Regression vs. Logistic Regression



In the figures, the orange marks indicate the **target Y (either 0 or 1)**



(a) The response variable is coded as 0 and 1 and a liner regression model can be conducted. But the **linear regression** might produce probabilities which are not bounded to [0, 1]. The should not be interpreted as probabilities directly

(b) **Logistic regression** ensures that our estimate for p(X) lies between 0 and 1

$$logistic\ (X) = p(X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}} = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

The relationship between $p(X)$ and X is not a straight line, but regardless of the value of X, if $\beta 1$ is positive then increasing X will be associated with increasing $p(X)$, and if $\beta 1$ is negative then increasing X will be associated with decreasing $p(X)$.

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

Logistic Regression Model

- **To build a logistic regression model**, we simply pass the output of the basic linear regression model through the logistic function

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = logistic(\mathbf{w} \cdot \mathbf{d}) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{d})}} = \frac{1}{1 + e^{-(\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \cdots + \mathbf{w}[m] \times \mathbf{d}[m])}} = \boldsymbol{P}(\mathbf{d})$$

  - $e \approx 2.71828$ is a mathematical constant [Euler's number.]

- **Training a logistic regression model**
  - Before we train a logistic regression model we map the binary target levels to 0 or 1
  - It is recommended that descriptive feature values always be normalized.
  - The training process uses a slightly modified version of the gradient descent algorithm

- The *error* of the model on each instance is the difference between the target feature (0 or 1) and the value of the prediction [0,1]

Estimating the Regression Parameters

- **Approaches for estimating the regression parameters**
  - Least Square Approach using Gradient Descent
  - Maximum Likelihood Approach

Gradient Descent for Logistic Regression Model

- The **weight update rule** (i.e., the **error delta function**) of the gradient descent algorithm is slightly changed for training logistic regression models.
- The new weight update rule for multivariable logistic regression is:

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \underbrace{\sum_{i=1}^{n}\left(\left(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)\right) \times \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \times \left(1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)\right) \times \mathbf{d}_i\,[j]\right)}_{\mathbf{errorDelta}(\mathcal{D},\, \mathbf{w}[j])}$$

  - For the detail, see pp344 – 346 in Kelleher's book

Training Illustration

- The **gradient descent algorithm** to minimize the sum of squared errors based on the training dataset for a selection of the logistic regression model.

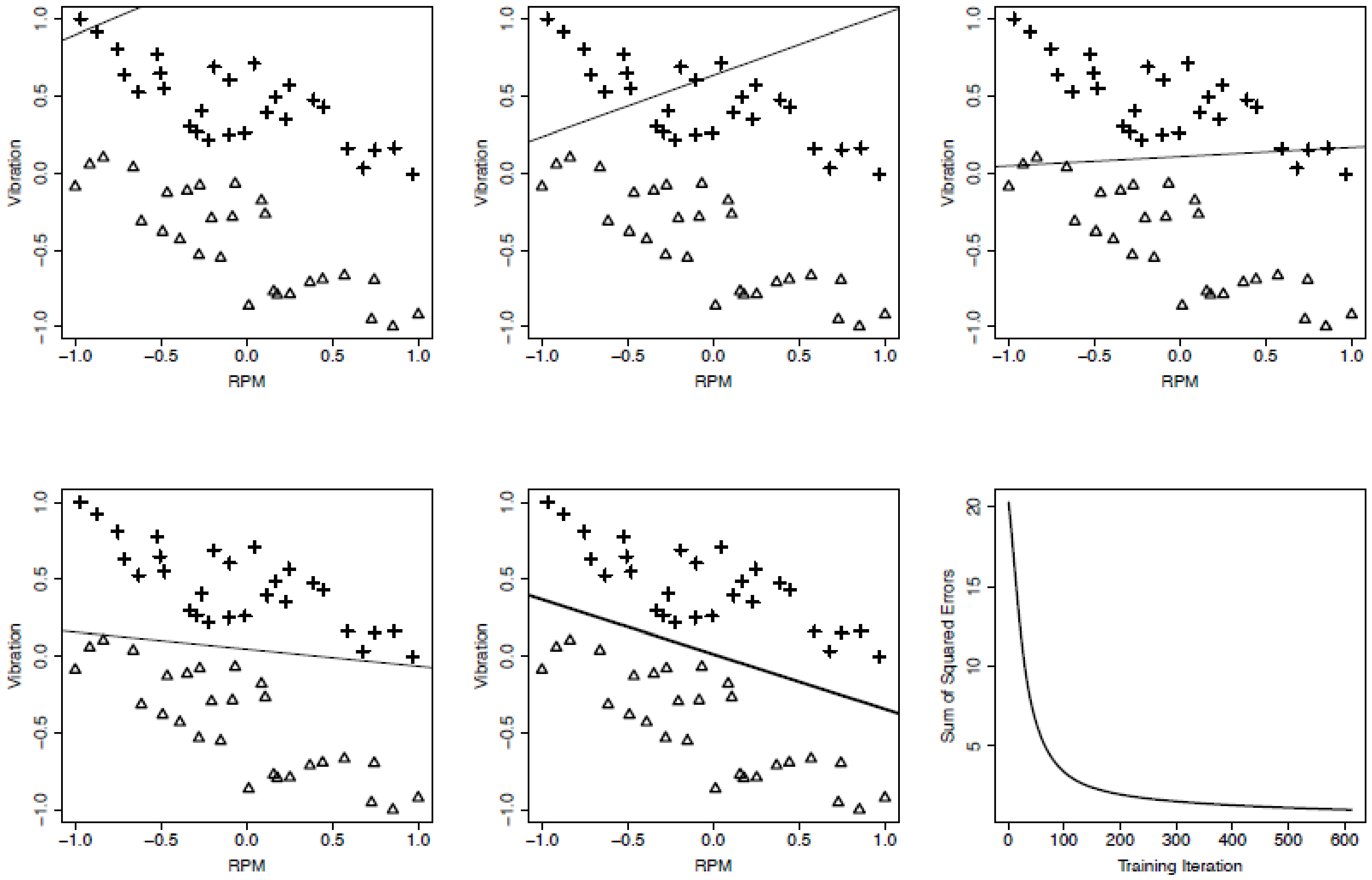


Figure: A selection of the logistic regression models developed during the gradient descent process for the generators dataset. The bottom-right panel shows the sum of squared error values generated during the gradient descent process.

Example: Logistic Regression Model

- Example: The resulting logistic regression model for the generator dataset is

$$\mathbb{M}_{\mathbf{w}}(\langle \text{RPM}, \text{VIBRATION} \rangle) = \frac{1}{1 + e^{-(-0.4077 + 4.1697 \times \text{RPM} + 6.0460 \times \text{VIBRATION})}}$$

- **Benefits** of Logistic Regression Model
  - There is a gentle transition from the predictions of the *faulty* target level to predictions of the *good* generator target level.
  - Logistic regression model outputs can be interpreted as probabilities of the occurrences of a target level

$$\boldsymbol{P}(t = \text{'faulty'} | \mathbf{d}) = \mathbb{M}_{\mathbf{w}}(\mathbf{d})$$

$$\boldsymbol{P}(t = \text{'good'} | \mathbf{d}) = 1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d})$$

    - The logical regression model typically outputs the probability that the instance belongs to the positive class (usually labeled ‘1’). In this case, faulty is converted to ‘1’.

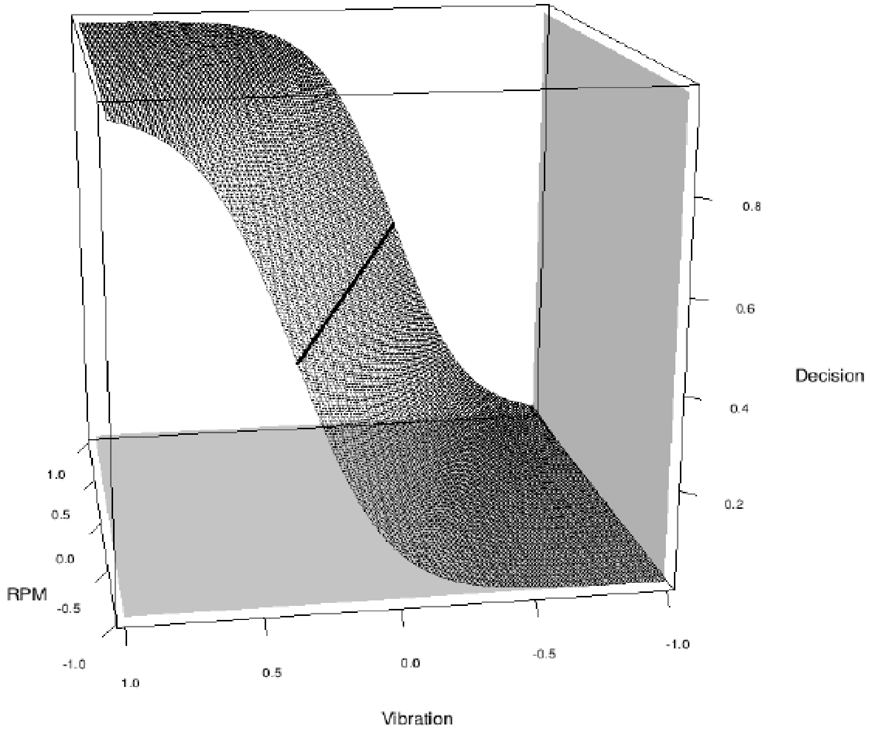


Figure: The logistic decision surface that results from training a model to represent the generators dataset

A Worked Example

ID	RPM	Vibration	Status	ID	RPM	Vibration	Status
1	498	604	faulty	35	501	463	good
2	517	594	faulty	36	526	443	good
3	541	574	faulty	37	536	412	good
4	555	587	faulty	38	564	394	good
5	572	537	faulty	39	584	398	good
6	600	553	faulty	40	602	398	good
7	621	482	faulty	41	610	428	good
8	632	539	faulty	42	638	389	good
9	656	476	faulty	43	652	394	good
10	653	554	faulty	44	659	336	good
11	679	516	faulty	45	662	364	good
12	688	524	faulty	46	672	308	good
13	684	450	faulty	47	691	248	good
14	699	512	faulty	48	694	401	good
15	703	505	faulty	49	718	313	good
16	717	377	faulty	50	720	410	good
17	740	377	faulty	51	723	389	good
18	749	501	faulty	52	744	227	good
19	756	492	faulty	53	741	397	good
20	752	381	faulty	54	770	200	good
21	762	508	faulty	55	764	370	good
22	781	474	faulty	56	790	248	good
23	781	480	faulty	57	786	344	good
24	804	460	faulty	58	792	290	good
25	828	346	faulty	59	818	268	good
26	830	366	faulty	60	845	232	good
27	864	344	faulty	61	867	195	good
28	882	403	faulty	62	878	168	good
29	891	338	faulty	63	895	218	good
30	921	362	faulty	64	916	221	good
31	941	301	faulty	65	950	156	good
32	965	336	faulty	66	956	174	good
33	976	297	faulty	67	973	134	good
34	994	287	faulty	68	1002	121	good

Table An extended generators dataset version

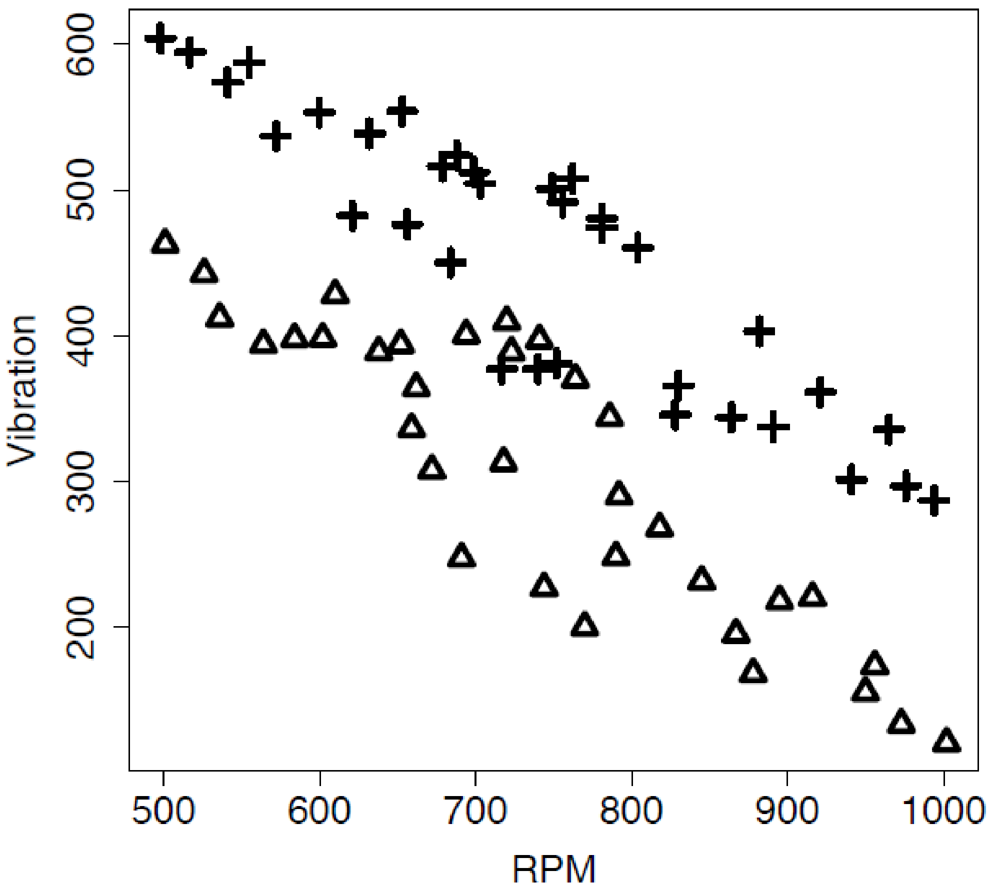


Figure: A scatter plot of the **extended generators dataset**, which results in instances with the different target levels overlapping with each other. 'good' generators are shown as crosses, and 'faulty' generators are shown as triangles.

Example: Training Setup

- In this example, before the training process begins, both descriptive features are normalized to the range [-1, 1].
- Let's assume that the learning rate $\alpha = 0.02$
- And the initial weights are supposed like

Initial Weights

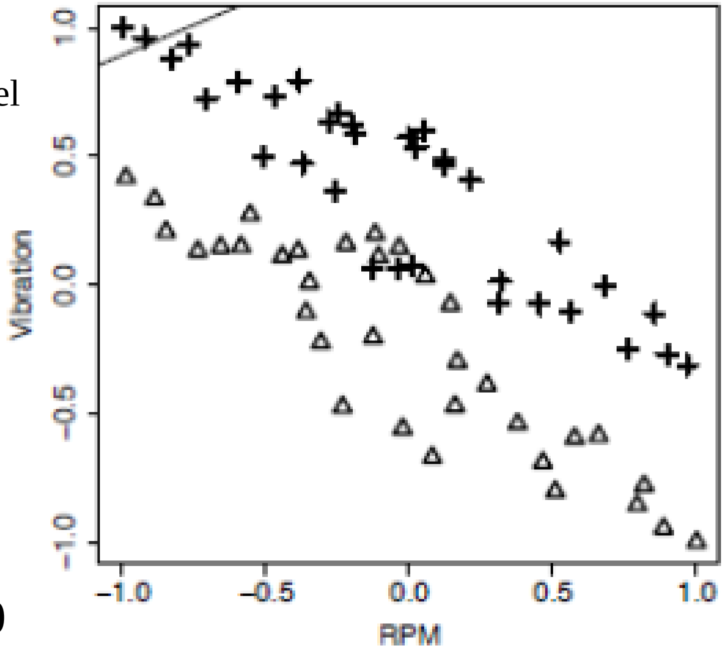
w[0]:	-2.9465	**w**[1]:	-1.0147	**w**[2]:	-2.1610

ID	RPM	VIBRATION	STATUS
1	498	604	faulty
2	517	594	faulty
3	541	574	faulty
⋮			

Data set

ID	TARGET LEVEL	Pred.
1	1	0.5570
2	1	0.5168
3	1	0.4469
4	1	0.4629
65	0	0.0037
66	0	0.0042
67	0	0.0028
68	0	0.0022

Figure. Only instance ID 1 and ID 2 (> 0.5) are predicted to Level 1 (cross), others are to Level 0 (triangle)



$$\mathbb{M}_{\mathbf{w}}(\text{ID1}) = \frac{1}{1+e^{-(\mathbf{w}[0]+\mathbf{w}[1]\times\mathbf{d}[1]+\mathbf{w}[2]\times\mathbf{d}[2])}} = \frac{1}{1+2.71828^{-(-2.9465\pm1.0147\times498\pm2.1610\times604)}} = 0.5570$$

Example: First Iteration – Weight Update

- Current weights:

Initial Weights

w[0]:	-2.9465	**w**[1]:	-1.0147	**w**[2]:	-2.1610

- Weight update: $\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \underbrace{\sum_{i=1}^{n}\left(\left(t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)\right) \times \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \times \left(1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)\right) \times \mathbf{d}_i\,[j]\right)}_{\text{errorDelta}(\mathcal{D},\, \mathbf{w}[j])}$
  - Calculate the delta value for each weight, errorDelta($\mathcal{D}$, w[j]) and then update the weight,
  - e.g., **w**[0]← -2.9465 + 0.02 × 2.7031 = 2.8924

Iteration 1

ID	Target Level	Pred.	Error	Squared Error	errorDelta($\mathcal{D}$, **w**[i]) **w**[0]	**w**[1]	**w**[2]
1	1	0.5570	0.4430	0.1963	0.1093	-0.1093	0.1093
2	1	0.5168	0.4832	0.2335	0.1207	-0.1116	0.1159
3	1	0.4469	0.5531	0.3059	0.1367	-0.1134	0.1197
4	1	0.4629	0.5371	0.2885	0.1335	-0.1033	0.1244
			...				
65	0	0.0037	-0.0037	0.0000	0.0000	0.0000	0.0000
66	0	0.0042	-0.0042	0.0000	0.0000	0.0000	0.0000
67	0	0.0028	-0.0028	0.0000	0.0000	0.0000	0.0000
68	0	0.0022	-0.0022	0.0000	0.0000	0.0000	0.0000
			Sum	24.4738	2.7031	-0.7015	1.6493
Sum of squared errors (Sum/2)				12.2369			

- New weights:

New Weights (after Iteration 1)

w[0]:	-2.8924	**w**[1]:	-1.0287	**w**[2]:	-2.1940

Example: Second Iteration

- Current weights:

Weights (after Iteration 1)					
w[0]:	-2.8924	**w**[1]:	-1.0287	**w**[2]:	-2.1940

- Weight update: $\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \text{errorDelta}(\mathcal{D}, \mathbf{w}[j])$
  - e.g., $\mathbf{w}[1] \leftarrow -1.0287 + 0.02 \times -0.6646 = -1.0416$

- New weights:

New Weights (after Iteration 2)					
w[0]:	-2.8380	**w**[1]:	-1.0416	**w**[2]:	-2.2271

Iteration 2

ID	Target Level	Pred.	Error	Squared Error	errorDelta($\mathcal{D}$, **w**[i]) **w**[0]	**w**[1]	**w**[2]
1	1	0.5817	0.4183	0.1749	0.1018	-0.1018	0.1018
2	1	0.5414	0.4586	0.2103	0.1139	-0.1053	0.1094
3	1	0.4704	0.5296	0.2805	0.1319	-0.1094	0.1155
4	1	0.4867	0.5133	0.2635	0.1282	-0.0992	0.1194
				...			
65	0	0.0037	-0.0037	0.0000	0.0000	0.0000	0.0000
66	0	0.0043	-0.0043	0.0000	0.0000	0.0000	0.0000
67	0	0.0028	-0.0028	0.0000	0.0000	0.0000	0.0000
68	0	0.0022	-0.0022	0.0000	0.0000	0.0000	0.0000
			Sum	24.0524	2.7236	-0.6646	1.6484
			Sum of squared errors (Sum/2)	12.0262			

Example: Final Model

- The final model found is:

$$\mathbb{M}_w(< \mathrm{RPM}, \mathrm{VIBRATION} >) = \frac{1}{1 + e^{-(-0.4077 + 4.1697 \times \mathrm{RPM} + 6.0460 \times \mathrm{VIBRATION})}}$$

Example: Training Illustration

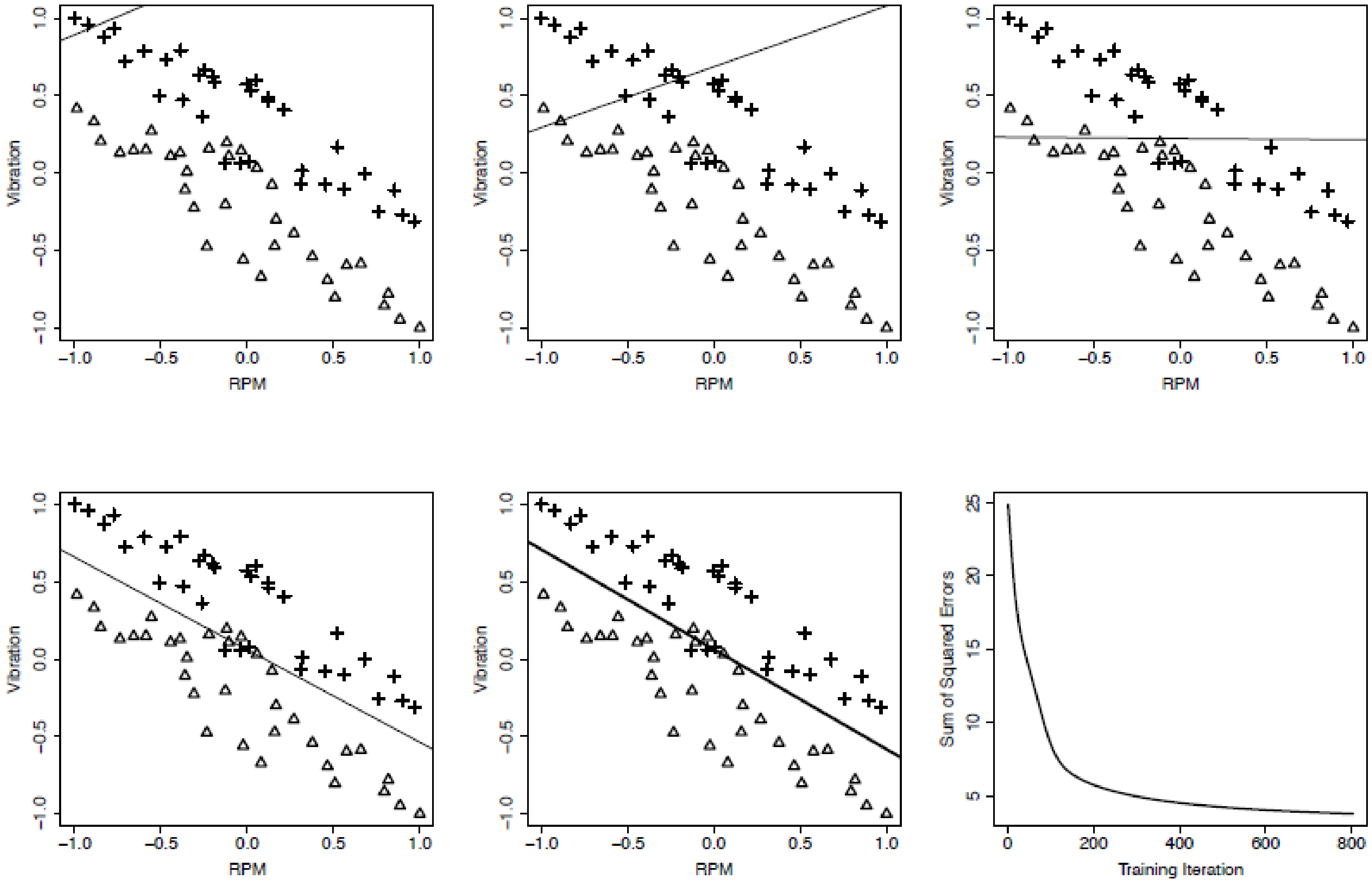


Figure: A selection of the logistic regression models developed during the gradient descent process for the extended generators dataset. The bottom-right panel shows the sum of squared error values generated during the gradient descent process.

Outline

- Fundamentals
- Standard Approach: Multivariable Linear Regression with Gradient Descent
- Extensions and Variations
  - Interpreting Multivariable Linear Regression Models
  - Assessing the Parameter Estimates
  - Setting the Learning Rate Using Weight Decay
  - Handling Categorical Descriptive Features
  - Handling Categorical Target Features: Logistic Regression
  - ☞ **Modeling Non-linear Relationships**
  - Multinomial Logistic Regression

Modeling Non-linear Relationships

- Linear models work well **when the underlying relationships in the data are linear**
- If the underlying data exhibit non-linear relationships, **how to model it?**

Example Dataset

ID	RAIN	GROWTH	ID	RAIN	GROWTH	ID	RAIN	GROWTH
1	2.153	14.016	12	3.754	11.420	23	3.960	10.307
2	3.933	10.834	13	2.809	13.847	24	3.592	12.069
3	1.699	13.026	14	1.809	13.757	25	3.451	12.335
4	1.164	11.019	15	4.114	9.101	26	1.197	10.806
5	4.793	4.162	16	2.834	13.923	27	0.723	7.822
6	2.690	14.167	17	3.872	10.795	28	1.958	14.010
7	3.982	10.190	18	2.174	14.307	29	2.366	14.088
8	3.333	13.525	19	4.353	8.059	30	1.530	12.701
9	1.942	13.899	20	3.684	12.041	31	0.847	9.012
10	2.876	13.949	21	2.140	14.641	32	3.843	10.885
11	4.277	8.643	22	2.783	14.138	33	0.976	9.876

Table: Grass Growth Dataset - A dataset describing grass growth on Irish farms during July 2012.

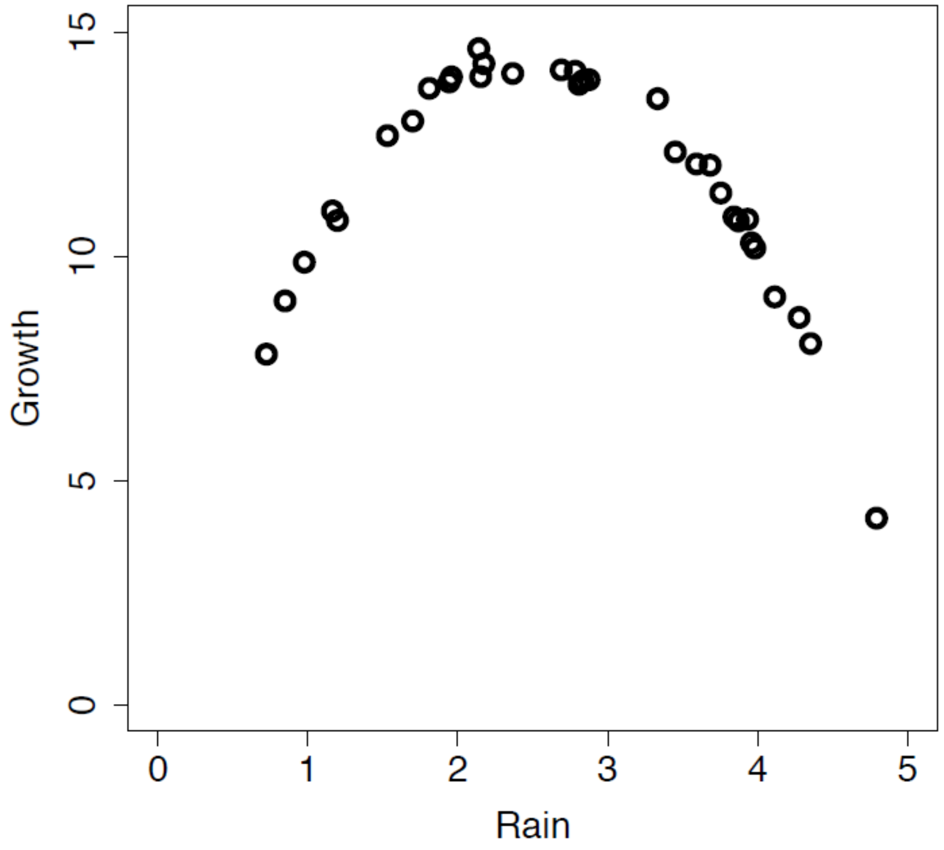


Figure: A scatter plot of the RAIN and GROWTH feature from the grass growth dataset. - **Strong non-linear relationship between rainfall and grass growth**

Non-Linear Relationships

- The best simple linear model we can learn for this data is: GROWTH = 13.510 + -0.667 × RAIN
- However, to successfully model the non-linear relationship, **non-linear elements should be introduced.**
- General approach is to introduce "**basis functions**" that transform the raw inputs to the model into non-linear representations but still keep the model itself linear in terms of the weights.
- The advantage of this is that, except for introducing the mechanism of basis functions, we do not need to make any other changes to the approach we have presented so far.

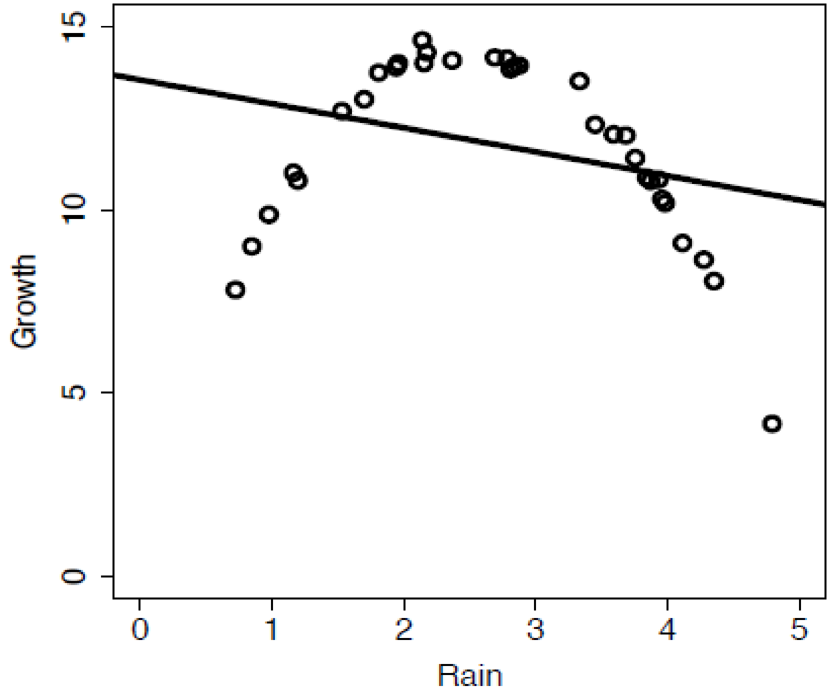


Figure: A simple linear regression model trained to capture the relationship between the grass growth and rainfall.

Linear Regression with Basis Functions

- The **simple linear regression model with basis functions**

$$\mathbb{M}_{w}(\mathbf{d}) = \sum_{k=0}^{b} \mathbf{w}[k] + \phi_k(\mathbf{d})$$

Where **w** is a set of b weights, and $\boldsymbol{\phi_0}$ **to** $\boldsymbol{\phi_b}$ **are a series of** $\boldsymbol{b}$ **basis functions** that each transform the input vector **d** in a different way.

Usually there are more basis functions than descriptive feature (i.e., $b > m$)

- Different kind of basis functions for polynomial and non- polynomial relationships
  - Quadratic functions
  - Cubic functions
  - Higher-degree polynomial functions
  - Spline functions
  - Radial bias functions,
  - and so on.

Linear Regression with Basis Functions

- The **simple linear regression model with basis functions**

$$\mathbb{M}_{w}(\mathbf{d}) = \sum_{k=0}^{b} \mathbf{w}[k] + \phi_k(\mathbf{d})$$

Where **w** is a set of *b* weights, and $\boldsymbol{\phi_0}$ **to** $\boldsymbol{\phi_b}$ **are a series of *b* basis functions** that each transform the input vector **d** in a different way.

Usually there are more basis functions than descriptive feature (i.e., $b > m$)

- One of the most common uses of basis functions in linear regression is to train models to capture **polynomial** relationship
  - The target is calculated from the descriptive features using only the addition of the descriptive feature values multiplied by weight values.
  - Polynomial relationships allow multiplication of descriptive feature values by each other and raising of descriptive features to exponents

Basis Function – Quadratic function for Polynomial Relationship

- The most common form of ***polynomial relationship*** is the **second order polynomial** (Quadratic), also known as the **quadratic function**
  - The general form is $a = bx \times cx^2$
- **Example**: The relationship between rainfall and grass growth in the grass growth dataset can be accurately represented as a second order polynomial

$$\textsc{Growth} = \mathbf{w}[0] \times \phi_0(\textsc{Rain}) + \mathbf{w}[1] \times \phi_1(\textsc{Rain}) + \mathbf{w}[2] \times \phi_2(\textsc{Rain})$$

, where $\boldsymbol{\phi_0}(\textbf{RAIN}) = \mathbf{1}$, $\boldsymbol{\phi_1}(\textbf{RAIN}) = \textbf{RAIN}$ **(linear term)**, $\boldsymbol{\phi_2}(\textbf{RAIN}) = \textbf{RAIN}^{\mathbf{2}}$ **(quadratic term)**

  - This model captures the non-linear relationship
  - But this model is still linear in terms of the weights and **can be trained using gradient descent**

Training Illustration

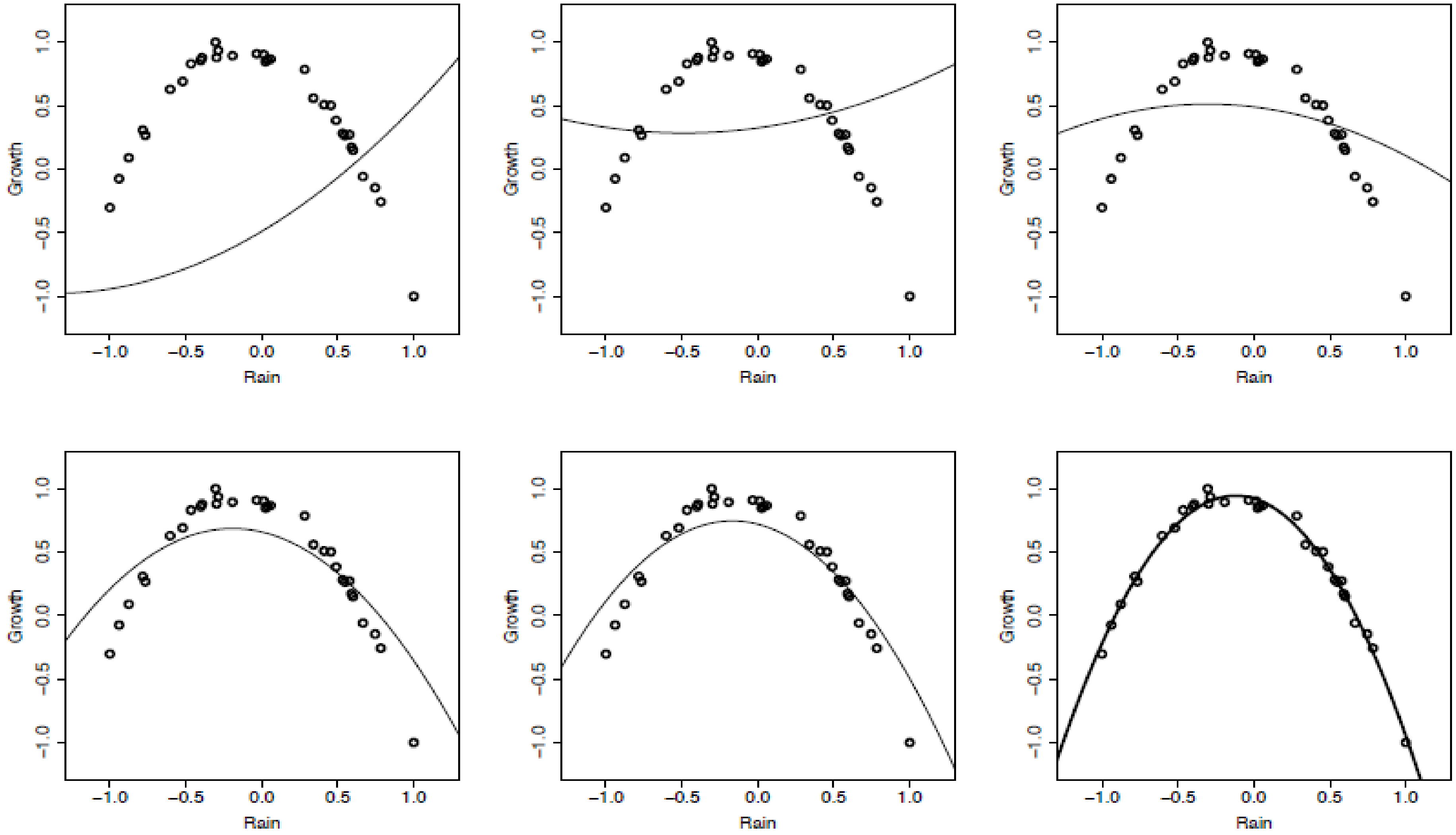


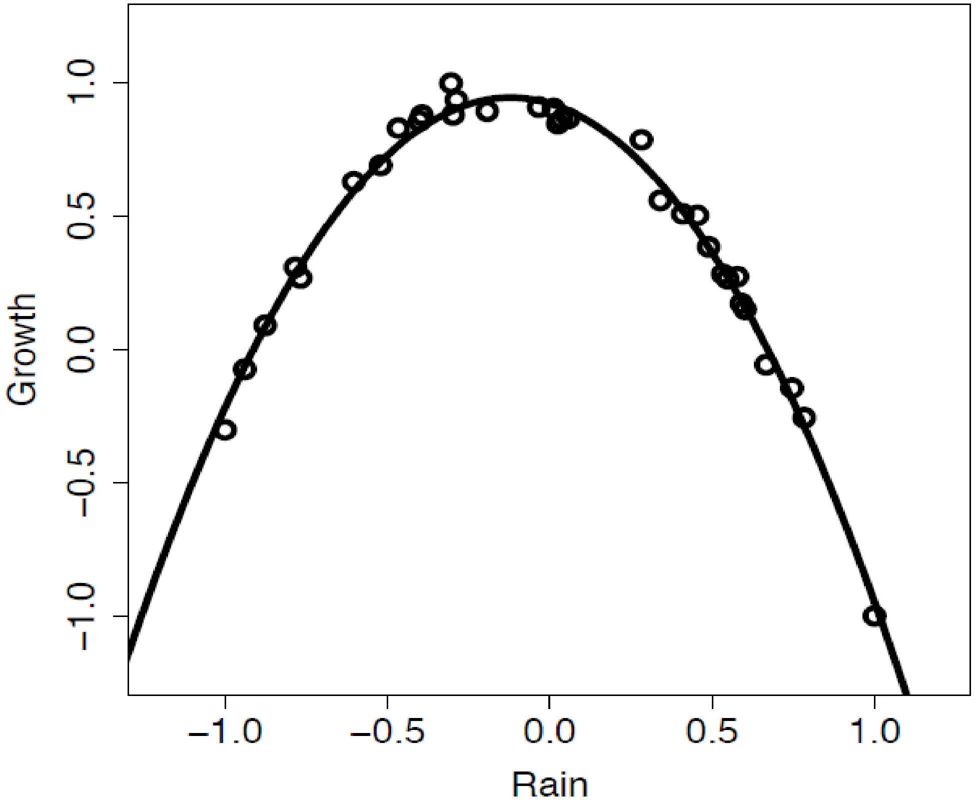
Figure: A selection of the models developed during the gradient descent process for the grass growth dataset. (Note that the RAIN and GROWTH features have been **range normalized** to the [-1, 1] range.)

Example: Final Model and Prediction

$$\boldsymbol{Growth = 0.3707 + \phi_0(Rain) + 0.8475 \times \phi_1(Rain) + (-1.717) \times \phi_2(Rain)}$$

- What is the predicted growth for the following RAIN values:

  1. RAIN= -0.75 => GROWTH = -1.2328
  2. RAIN= 0.1 => GROWTH = 0.43828
  3. RAIN= 0.9 => GROWTH = -0.25888



Non-linear Relationship with Categorical Target Feature

- Basis functions can also be used for multivariable simple linear regression models in the same way
- The only extra requirement **being the definition of more basis functions to train logistic regression models** for categorical prediction problems that involve non-linear relationships.

Example Dataset

ID	P20	P45	TYPE	ID	P20	P45	TYPE
1	0.4497	0.4499	negative	26	0.0656	0.2244	positive
2	0.8964	0.9006	negative	27	0.6336	0.2312	positive
3	0.6952	0.3760	negative	28	0.4453	0.4052	positive
4	0.1769	0.7050	negative	29	0.9998	0.8493	positive
5	0.6904	0.4505	negative	30	0.9027	0.6080	positive
6	0.7794	0.9190	negative	31	0.3319	0.1473	positive
		⋮				⋮	

Table: EEG (electroencephalograph) dataset - A dataset showing participants' responses to viewing 'positive' and 'negative' images measured on the EEG P20 and P45 potentials.

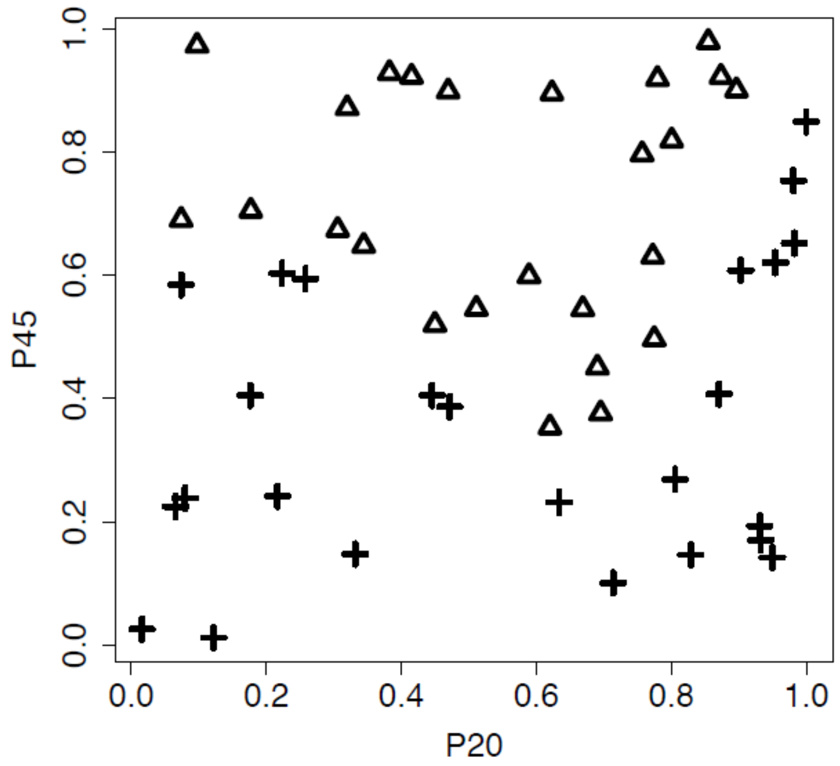


Figure: A scatter plot of from the EEG dataset. 'positive' images are shown as crosses, and 'negative' images are shown as triangles.

The two types of images are no linearly separable

Logistic Regression Model with Basis Functions

- The non-linear decision boundary can be represented using a **third-order polynomial** (Cubic) in the two descriptive features.
- A **logistic regression model using basis functions** is defined as follows:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \frac{1}{1 + e^{-\left(\sum_{j=0}^{b} \mathbf{w}[j]\phi_j(\mathbf{d})\right)}}$$

- We will use the following basis functions for the EEG problem:

$$\begin{aligned}
\phi_0(\langle P20, P45\rangle) &= 1 & \phi_4(\langle P20, P45\rangle) &= P45^2\\
\phi_1(\langle P20, P45\rangle) &= P20 & \phi_5(\langle P20, P45\rangle) &= P20^3\\
\phi_2(\langle P20, P45\rangle) &= P45 & \phi_6(\langle P20, P45\rangle) &= P45^3\\
\phi_3(\langle P20, P45\rangle) &= P20^2 & \phi_7(\langle P20, P45\rangle) &= P20 \times P45
\end{aligned}$$

Training Illustration

- The model can be trained **using gradient descent**

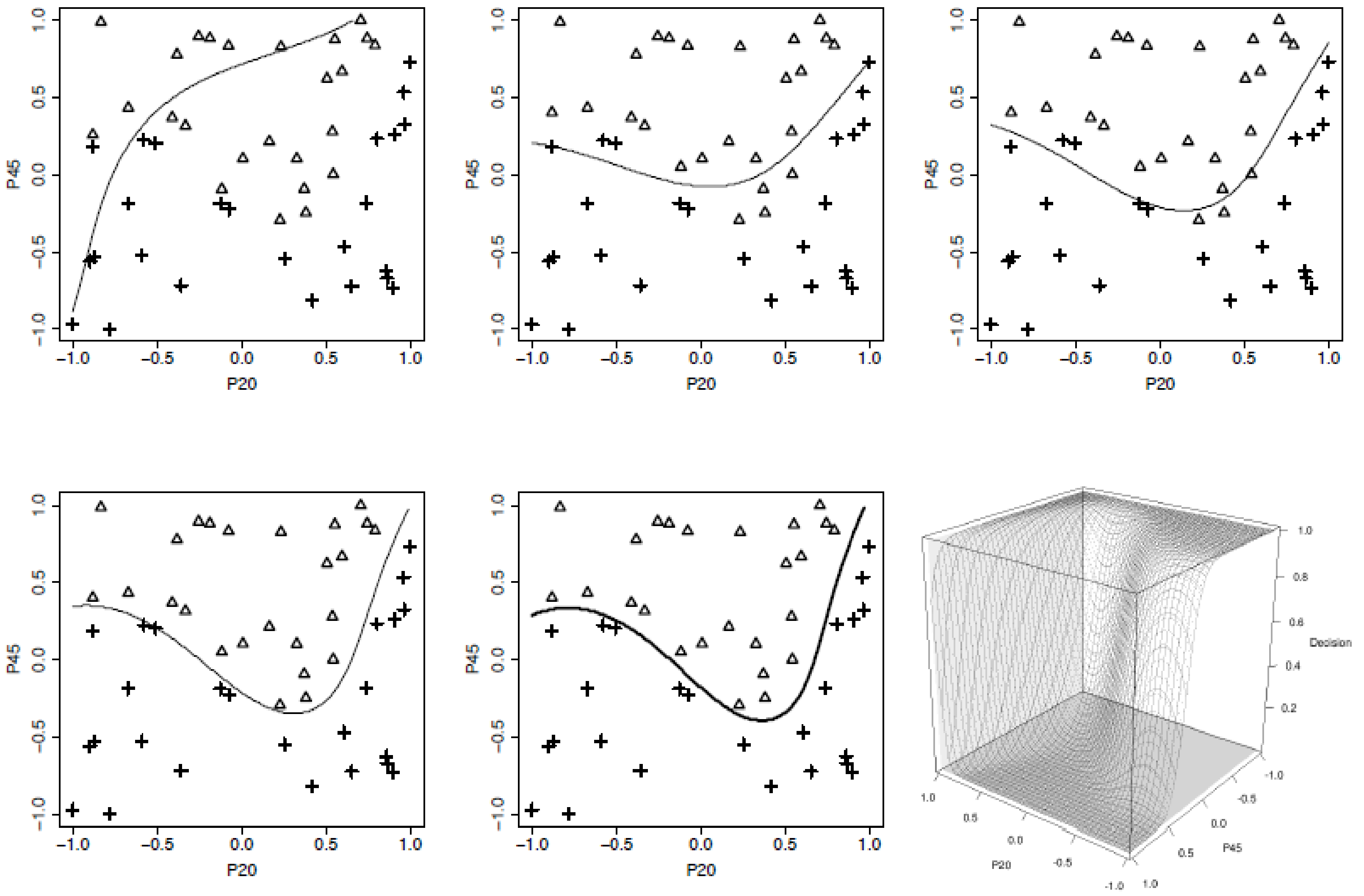


Figure: A selection of the models developed during the gradient descent process for the EEG dataset]. The final panel shows the decision surface generated.

Characteristics of Basis Function Usage

- **Advantages** of using basis functions
  - Using basis functions is a simple and effective way in which <u>to capture non-linear relationships within a linear regression model.</u>
  - There is no limit to the kinds of functions that can be used as basis functions
- **Disadvantage** of using basis functions
  - <u>The analyst has to design the basis function set </u>that will be used, although there are some well-known sets of functions—for example, different order polynomial functions—this can be a considerable challenge.
  - As the number of basis functions grows beyond the number of descriptive features, <u>the complexity of our models increases</u>, so the gradient descent process must search through a more complex weight space.

Outline

- Fundamentals
- Standard Approach: Multivariable Linear Regression with Gradient Descent
- Extensions and Variations
  - Interpreting Multivariable Linear Regression Models
  - Assessing the Parameter Estimates
  - Setting the Learning Rate Using Weight Decay
  - Handling Categorical Descriptive Features
  - Handling Categorical Target Features: Logistic Regression
  - Modeling Non-linear Relationships
  - ☞ **Multinomial Logistic Regression**

Multinomial Logistic Regression

- The **multinomial logistic regression model** is an extension that handles categorical target features with more than two levels.
- A good way to build multinomial logistic regression models is to use a set of **one-versus-all models**.
  - If we have r target levels, we create r one-versus-all logistic regression models.
  - A one-versus-all model distinguishes between one level of the target feature and all the others.

Example Dataset

ID	SPEND	FREQ	TYPE
1	21.6	5.4	single
2	25.7	7.1	single
3	18.9	5.6	single
4	25.7	6.8	single
		⋮	
26	107.9	5.8	business
27	92.9	5.5	business

ID	SPEND	FREQ	TYPE
28	122.6	6.0	business
29	107.7	5.7	business
		⋮	
47	53.2	2.6	family
48	52.4	2.0	family
49	46.1	1.4	family
50	65.3	2.2	family

Table: Customer type dataset - A dataset of customers of a large national retail chain.

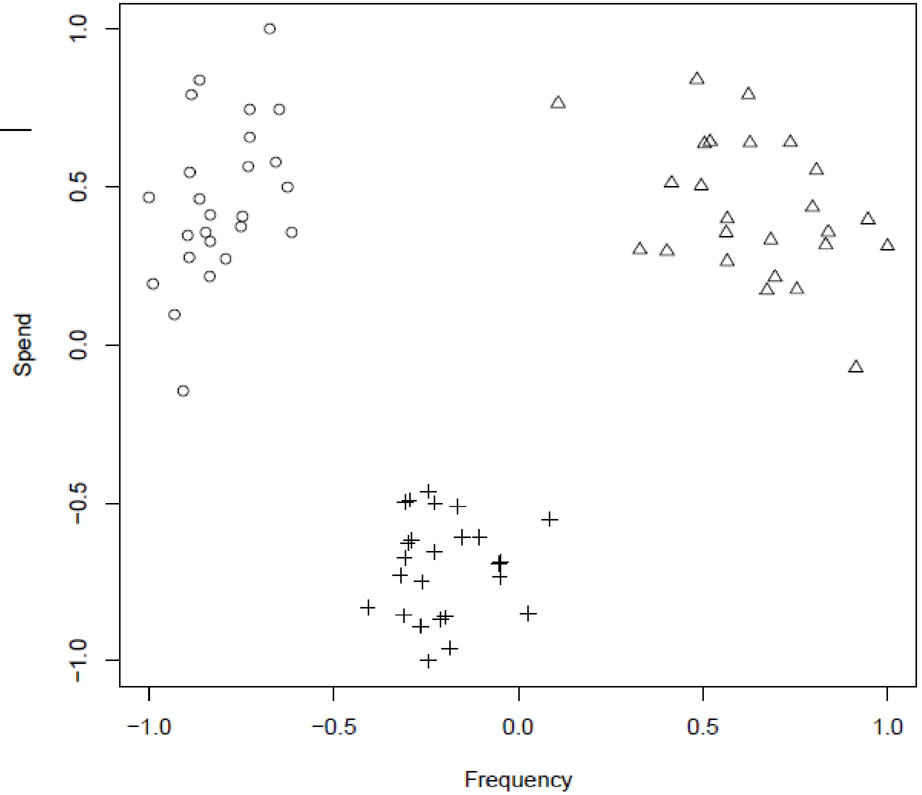
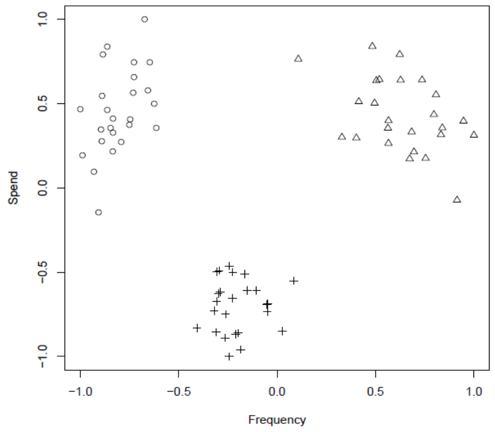


Figure: A scatter plot of from the customer type dataset that has three target levels ’single’ (squares), ’business’ (triangles) and ’family’ (crosses).

Example: Different One-Versus-All Prediction Models



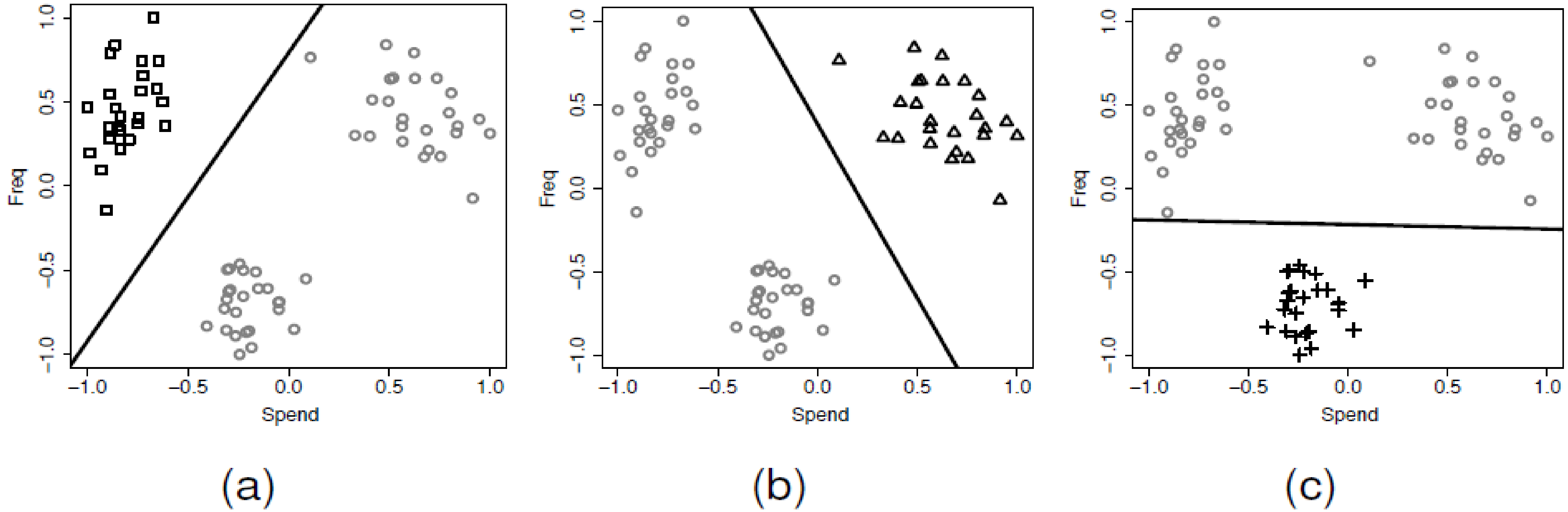


Figure: An illustration of three different **one-versus-all** prediction models for the customer type dataset] that has three target levels ’single’ (squares), ’business’ (triangles) and ’family’ (crosses).

Multinomial Logistic Regression Model

- For r target feature levels, we build r separate logistic regression models $\mathbb{M}_{\mathbf{w_1}}$ to $\mathbb{M}_{\mathbf{w_r}}$:

$$\begin{aligned}\mathbb{M}_{\mathbf{w_1}}(\mathbf{d}) &= logistic(\mathbf{w_1}\cdot\mathbf{d})\\ \mathbb{M}_{\mathbf{w_2}}(\mathbf{d}) &= logistic(\mathbf{w_2}\cdot\mathbf{d})\\ &\vdots\\ \mathbb{M}_{\mathbf{w_r}}(\mathbf{d}) &= logistic(\mathbf{w_r}\cdot\mathbf{d})\end{aligned}$$

, where $\mathbb{M}_{\mathbf{w_1}}$ to $\mathbb{M}_{\mathbf{w_r}}$ are r different one-versus-all logistic regression models, and $\mathbf{w_1}$ to $\mathbf{w_r}$ are r different sets of weights.

- To combine the outputs of these different models, we **normalize** their results using:

$$\mathbb{M}'_{\mathbf{w_k}}(\mathbf{d}) = \frac{\mathbb{M}_{\mathbf{w_k}}(\mathbf{d})}{\sum_{l\in levels(t)}\mathbb{M}_{\mathbf{w_l}}(\mathbf{d})}$$

  - where $\mathbb{M}_{\mathbf{w}_k}(\mathbf{d})$ is the one-versus-all model of the target level k.

and $\mathbb{M}'_{\mathbf{w}_k}(\mathbf{d})$ is a revised, normalized prediction for the one-versus-all model for the target level k.

- The ***sum of squared errors*** for each model is $\boldsymbol{SSE}(\mathbb{M}_{\mathbf{w}_k},\mathcal{D}) = \frac{1}{2}\sum_{i=1}^{n}\left(t_i - \mathbb{M}'_{\mathbf{w}_k}(d_i)\right)^2$

Training Illustration

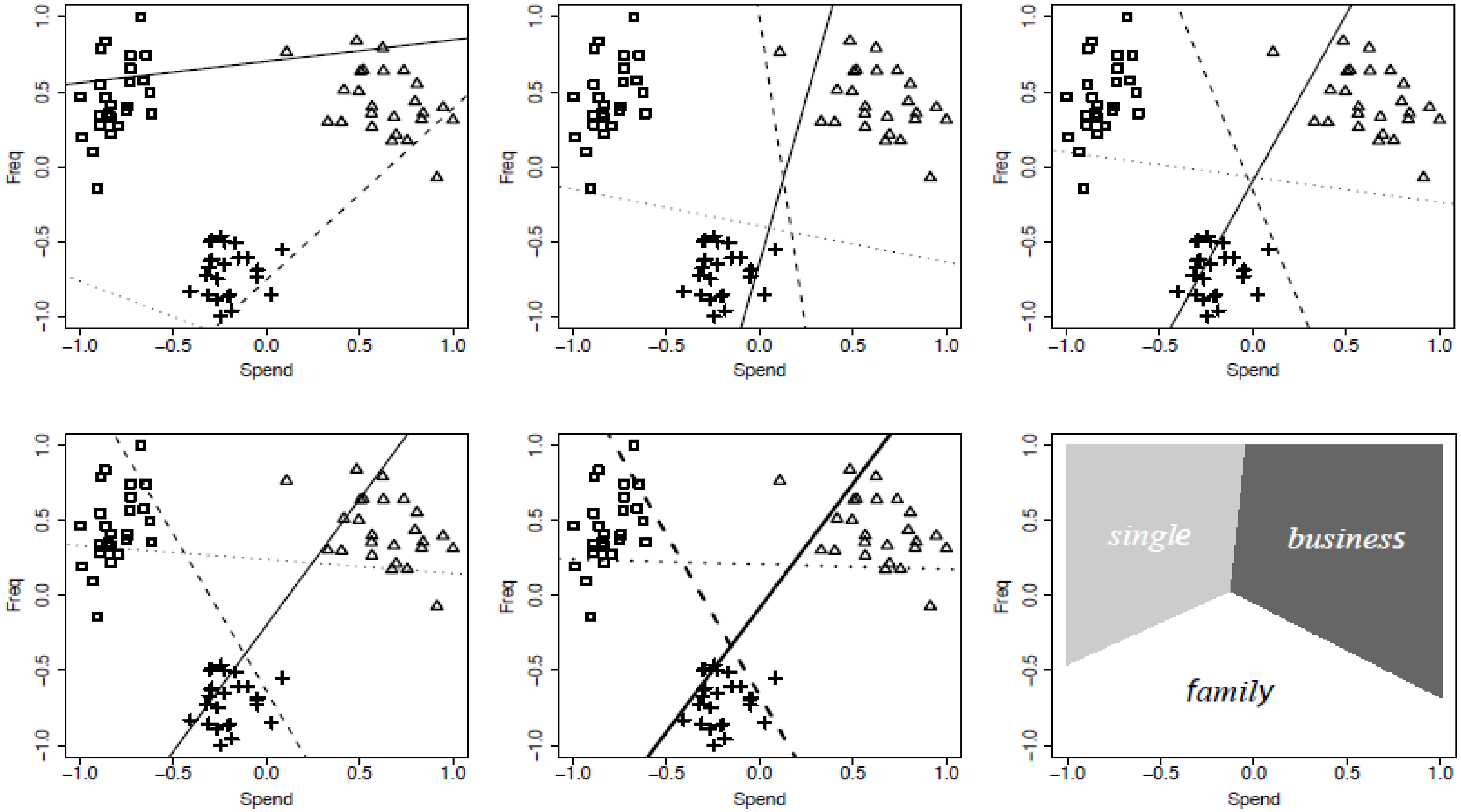
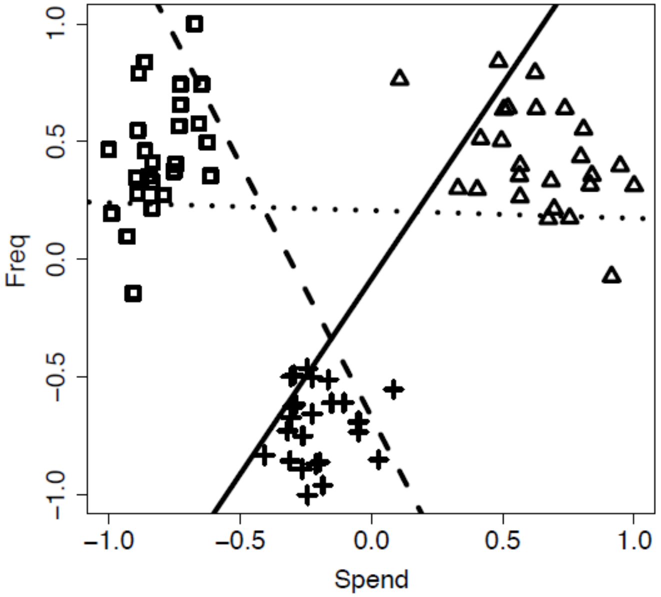


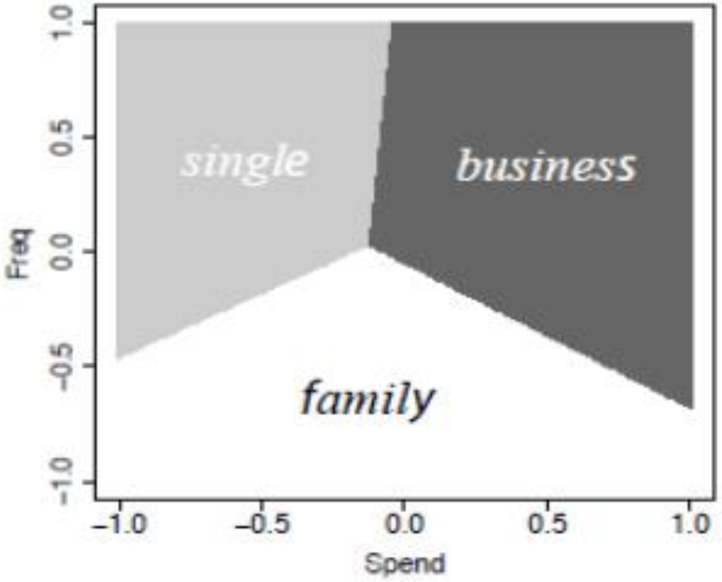
Figure: A selection of the models developed during the gradient descent process for the customer group dataset. Squares represent instances with the ’single’ target level, triangles the ’business’ level and crosses the ’family’ level. (f) illustrates the overall decision boundaries that are learned between the three target levels.

Final Model and Class Prediction

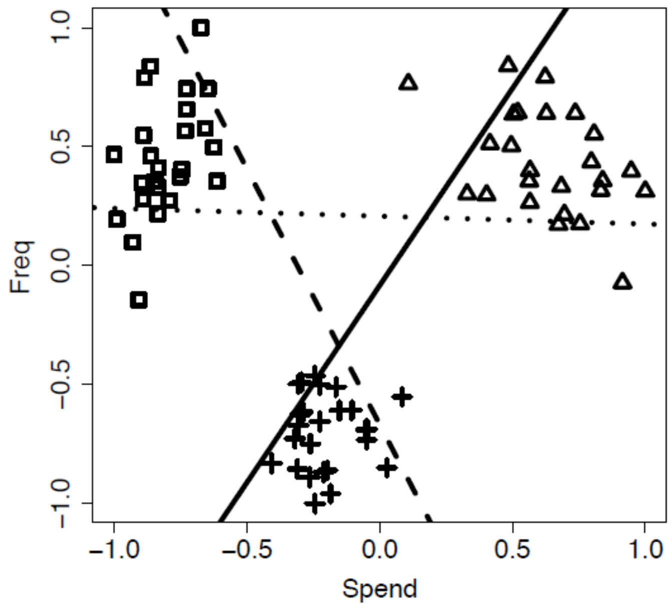
- The predicted level for a query, **q**, is the level associated with the one-versus-all model that outputs the highest result after normalization.

$$\mathbb{M}(\mathbf{q}) = \underset{l \in levels(t)}{\operatorname{argmax}} \; \mathbb{M}'_{\mathbf{w}_l}(\mathbf{q})$$





Example



- **Trained Model**

$$\mathbb{M}_{\mathbf{w}_{'single'}}(\mathbf{q}) = Logistic(0.7993 - 15.9030 \times \text{SPEND} + 9.5974 \times \text{FREQ})$$

$$\mathbb{M}_{\mathbf{w}_{'family'}}(\mathbf{q}) = Logistic(3.6526 + -0.5809 \times \text{SPEND} - 17.5886 \times \text{FREQ})$$

$$\mathbb{M}_{\mathbf{w}_{'business'}}(\mathbf{q}) = Logistic(4.6419 + 14.9401 \times \text{SPEND} - 6.9457 \times \text{FREQ})$$

- For **a query instance** with SPEND = 25.67 and FREQ = 6.12, which **are normalized to** SPEND = - 0.7279 and FREQ = 0.4789, **the predictions of the individual models** would be

$$\begin{aligned}\mathbb{M}_{\mathbf{w}_{'single'}}(\mathbf{q}) &= Logistic(0.7993 - 15.9030 \times (-0.7279) + 9.5974 \times 0.4789)\\ &= 0.9999\\ \mathbb{M}_{\mathbf{w}_{'family'}}(\mathbf{q}) &= Logistic(3.6526 + -0.5809 \times (-0.7279) - 17.5886 \times 0.4789)\\ &= 0.01278\\ \mathbb{M}_{\mathbf{w}_{'business'}}(\mathbf{q}) &= Logistic(4.6419 + 14.9401 \times (-0.7279) - 6.9457 \times 0.4789)\\ &= 0.0518\end{aligned}$$

Example (cont.)

- These predictions would be **normalized** as follows using $\mathbb{M}'_{\mathbf{w_k}}(\mathbf{d}) = \frac{\mathbb{M}_{\mathbf{w_k}}(\mathbf{d})}{\sum_{l \in levels(t)} \mathbb{M}_{\mathbf{w}_l}(\mathbf{d})}$

$$\mathbb{M}'_{\mathbf{w}_{'single'}}(\mathbf{q}) = \frac{0.9999}{0.9999 + 0.01278 + 0.0518} = 0.9393$$

$$\mathbb{M}'_{\mathbf{w}_{'family'}}(\mathbf{q}) = \frac{0.01278}{0.9999 + 0.01278 + 0.0518} = 0.0120$$

$$\mathbb{M}'_{\mathbf{w}_{'business'}}(\mathbf{q}) = \frac{0.0518}{0.9999 + 0.01278 + 0.0518} = 0.0487$$

- Using $\mathbb{M}(\mathbf{q}) = \operatorname*{argmax}_{l \in levels(t)} \mathbb{M}'_{\mathbf{w}_l}(\mathbf{q})$, the overall prediction for the query instance is 'single', as this gets the highest normalized score.

Outline

- Fundamentals
- Standard Approach: Multivariate Linear Regression with Gradient Descent
- Extensions and Variations

☞ **Summary**

Summary

- The **simple multivariable linear regression** model makes a prediction for a continuous target feature based on a weighted sum of the values of a set of descriptive features.

$$\mathbb{M}_{w}(\mathbf{d}) = \mathbf{w} \cdot \mathbf{d} = \sum_{j=0}^{m} \mathbf{w}[j] \times \mathbf{d}[\boldsymbol{j}]$$

- In an error-based model, learning equates to finding the optimal values for these weights.
- Each of the infinite number of possible combinations of values for the weights will result in a model that fits, to some extent, the relationship present in the training data between the descriptive features and the target feature.
- The optimal values for the weights are the values that define the model with the minimum prediction error.

Summary (cont.)

- We use an **error function** (**sum of squared errors**) to measure how well a set of weights fits the relationship in the training data.
- The value of the error function for every possible weight combination defines an error surface —for each combination of weight values, we get a point on the surface whose coordinates are the weight values, with an elevation defined by the error of the model using the weight values.

Summary (cont.)

- To find the optimal set of weights,
  - we begin with a set of random weight values that corresponds to some random point on the error surface.
  - We then iteratively make small adjustments to these weights based on the output of the error function, which leads to a journey down the error surface that eventually leads to the optimal set of weights.
  - To ensure that we arrive at the optimal set of weights at the end of this journey across the error surface, we need to ensure that each step we take moves downward on the error surface.
  - We do this by directing our steps according to the **gradient** of the error surface at each step. This is the **gradient descent algorithm**

Summary (cont.)

- The simple multivariable linear regression model can be extended in many ways
- **Logistic regression** models allow us to predict categorical targets rather than continuous ones by placing a threshold on the output of the simple multivariable linear regression model using the logistic function.
- The simple linear regression and logistic regression models were only capable of representing **linear relationships** between descriptive features and a target feature.
- By applying a set of **basis functions** to descriptive features, models that represent **non-linear relationships** can be created.

Summary (cont.)

- The advantages of using basis functions is that they allow models that represent non-linear relationships to be built even though these models themselves remain a linear combination of inputs. Consequently, we can still use the gradient descent process to train them.
- The main disadvantages of using basis functions are, first, that we must manually decide what set of basis functions to use; and second, that the number of weights in a model using basis functions is usually far greater than the number of descriptive features, so finding the optimal set of weights involves a search across a much larger set of possibilities—that is, a much larger **weight space**.
- It is recommended that simple linear models be evaluated first and basis functions introduced only when the performance of the simpler models is deemed unsatisfactory

Summary (cont.)

- In order to handle categorical target features with more than two levels, that is **multinomial prediction** problems, we need to use a **one-versus-all approach** in which multiple models are trained.
  - This introduces something of an explosion in the number of weights required for a model, as we have an individual set of weights for every target feature level.
  - This is one reason that other approaches are often favored over logistic regression for predicting categorical targets with many levels.
- For the regression models, there is a large body of research and best practice in statistics. There is a range of techniques that allow a degree of analysis of regression models beyond what is possible for other approaches.