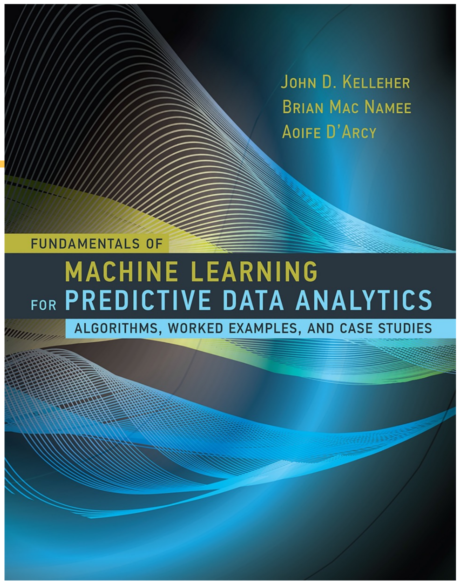
Topic 6


John D. Kelleher
Brian Mac Namee
Aoife D'Arcy
Fundamentals of
Machine Learning
for Predictive Data Analytics
Algorithms, Worked Examples, and Case Studies

Regression Analysis

[John D. Kelleher *et al*. 2015. *Fundamentals of Machine Learning for Predictive Data Analytics*. The MIT Press. ISBN: 0262029448.]

Contents

Introduction – Basic Concepts

- Notations used are

- a dataset D contains m instances: $D = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \ldots, m\}$

- an ith instance $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{id})$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, d$. When $d = 1$, $\mathbf{x}_i = (1, x_{i1})$.

- an instance $\mathbf{x} = (1, x_1, x_2, \ldots, x_d)$.

- target variable: y_i, y, or $t \in \{0, 1\}$

- predicted value: $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$, y_i', y'

- $\mathbf{w} = (w_0, w_1, \ldots, w_d)$ is called the **weight vector**. When $d = 1$, $\mathbf{w} = (w_0, w_1)$.

Introduction – Basic Concepts

- Notations used are

- **dot product**:

$$\mathbf{w} \cdot \mathbf{x} = w_0 x_0 + w_1 x_1 + \ldots + w_d x_d,$$

where a dummy attribute $x_0 = 1$.

$$\mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{d} w_i x_i + w_0$$

$$\mathbf{w} \cdot \mathbf{x} = \sum_{i=0}^{d} w_i x_i \text{ , where } x_0 = 1$$

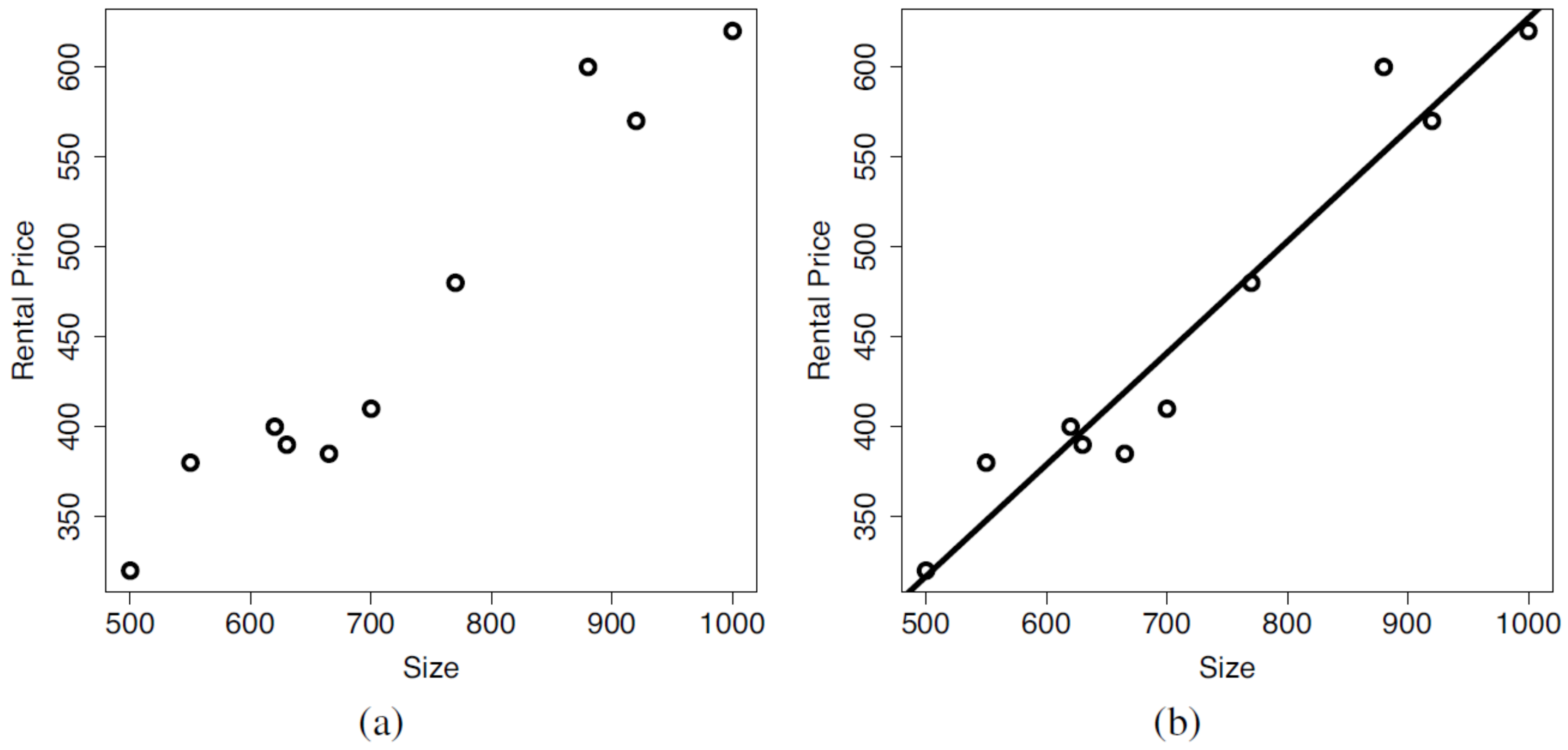
Introduction – Basic Concepts

- Given the office rentals dataset *D* shown in Table 7.1 below. We want to predict the **target variable** y = RENTAL PRICE for the descriptive feature $\mathbf{x}_{i1}$ = SIZE that we have never actually seen in the historical data.

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

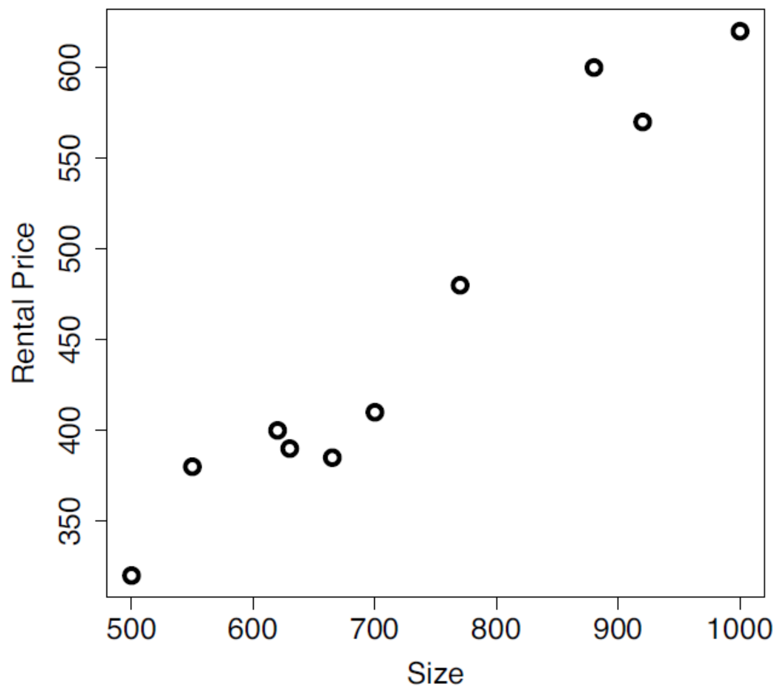
Introduction – Basic Concepts

• A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset *D* is shown in Fig. 7.1 below.

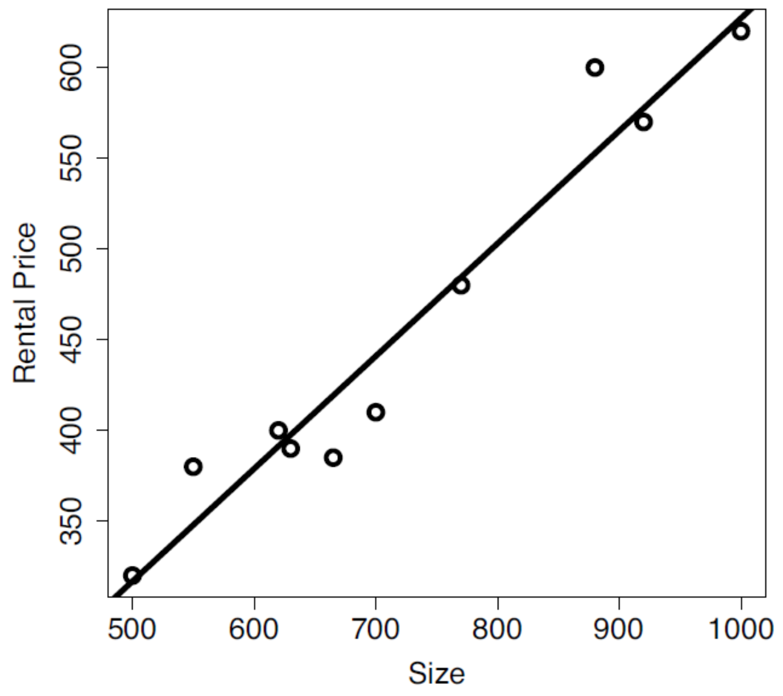


Introduction – Basic Concepts

- The plot shows the **linear relationship** between the two features SIZE and RENTAL PRICE.



(a)



(b)

1. Univariate Linear Regression (ULR)

- **Univariate linear regression** (ULR) model can be used to predict the **target** (**response**) **variable** y = RENTAL PRICE using the attribute $\mathbf{x}_{i1}$ = SIZE.
- That is, we can use the ULR model

$$\text{RENTAL PRICE} = w_0 + w_1 \times \text{SIZE}$$

$$(\text{e.g., RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE})$$

where w_1 is the **slope** of the line, w_0 is the ***y*-intercept** of the line, w_0 and w_1 are called **weights** or **regression coefficients**. (a.k.a. **model parameters**).

1. Univariate Linear Regression (ULR)

RENTAL PRICE = 6.47 + 0.62 × SIZE (7.2)

- We can use the above ULR model to predict the expected rental price of the 730 square foot office (i.e., SIZE = 730) by plugging the value of SIZE = 730 into the ULR model. That is, we have RENTAL PRICE = 6.47 + 0.62 × 730 = 459.07.

1. Univariate Linear Regression (ULR)

- For an instance $\mathbf{x}$ in a dataset D, the ULR model can be defined as

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} = w_0 + w_1 \times x_1, \; (7.3)$$

where

$h_{\mathbf{w}}(\mathbf{x}) = y' = h(\mathbf{x})$ is the prediction output by the model $h_{\mathbf{w}}(\mathbf{x})$ for the instance $\mathbf{x}$, $\mathbf{w} = (w_0, w_1)$, $\mathbf{x} = (1, x_1)$. We have $\mathbf{w} \cdot \mathbf{x} = w_0 x_0 + w_1 x_1$, $x_0 = 1$.

- That is, for an instance $\mathbf{x}_i = (1, x_{i1})$ in a dataset D (i.e., $d = 1$), we have $h_{\mathbf{w}}(\mathbf{x}_i) = w_0 + w_1 x_{i1}$.

Optimal Weights

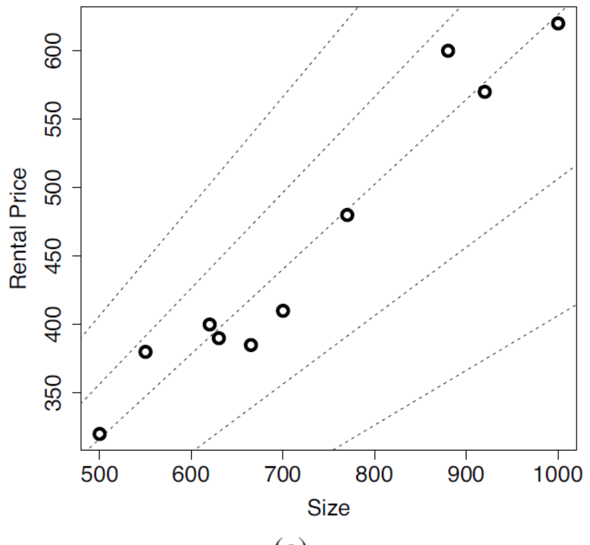
- Training a ULR models is determining the **optimal weights** in the model.
- The **optimal weights** are the ones that allow the model to best capture the relationship between the descriptive features $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{id})$ and a target feature y_i (prediction errors: $h_{\mathbf{w}}(\mathbf{x}_i) - y_i$).
- A set of weights $\mathbf{w} = (w_0, w_1, \ldots, w_d)$ that captures this relationship well are said to **fit** the training data.

1. Univariate Linear Regression (ULR)

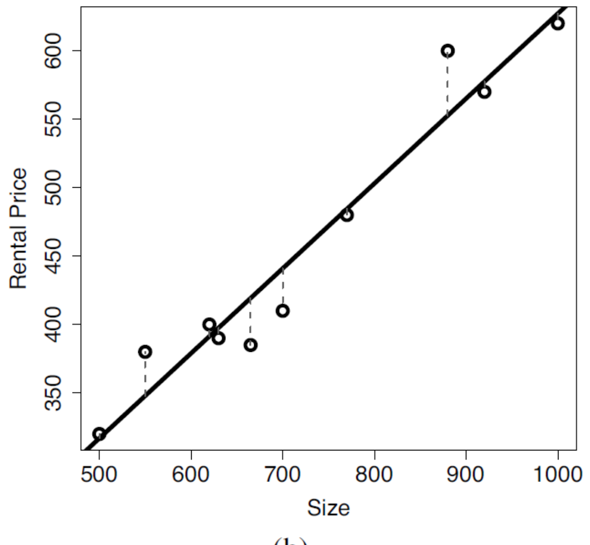
- In order to find the **optimal set of weights**, we need some way to measure how well a model is defined by using a candidate set of weights that fits a training dataset.
- We define an **error** (a.k.a. **lost** or **cost**) **function** to measure the error between the predictions y_i' a model makes based on the descriptive features for each instance $\mathbf{x}_i = (1, x_{i1}, x_{i2}, ..., x_{id})$ in D and the **actual target** values y_i for each instance in D. That is, prediction errors $e_i = y_i' - y_i$ (i.e., $e_i = h_{\mathbf{w}}(\mathbf{x}_i) - y_i$).

Measuring Error

- A collection of possible ULR models capturing the relationship between SIZE and RENTAL PRICE.



(a)



(b)

- The solid line (w_0 is fixed at 6.47, $w_1 = 0.62$) is the one that most accurately fits the training data D.

Error Function

- An **error function** captures the error between the predictions $h_{\mathbf{w}}(\mathbf{x}_i) = y_i'$ made by a model and the actual values y_i in D (i.e., $e_i = h_{\mathbf{w}}(\mathbf{x}_i) - y_i$).
- The most commonly used **error functions** are the **sum of squared errors** (SSE) L_2, mean squared errors (MSE), and mean absolute error (MAE).

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}_i) - y_i)^2, \ \mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m}|h(\mathbf{x}_i) - y_i|$$

Error Function

- Notice that the model sometimes over estimates the office rental price, and sometimes underestimates the office rental price. Thus, the prediction errors $e_i = h_{\mathbf{w}}(\mathbf{x}_i) - y_i$ can be positive or negative.
- If we were to simply add these errors together, the positive and negative errors would effectively cancel each other out.

Error Function

- The sum of squared errors (SSE) **error function** L_2 is defined as

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 \quad \textbf{(7.4)}$$

where D contains m training instances $\mathbf{x}_i$, each $\mathbf{x}_i$ has a target feature y_i, $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i = y_i'$ is the prediction made by a candidate model $h_{\mathbf{w}}$ for $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{id})$, and $h_{\mathbf{w}}$ is defined by the **weight vector** $\mathbf{w} = (w_0, w_1, \ldots, w_d)$ for $d = 1$.

Error Function

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 \quad \textbf{(7.4)}$$

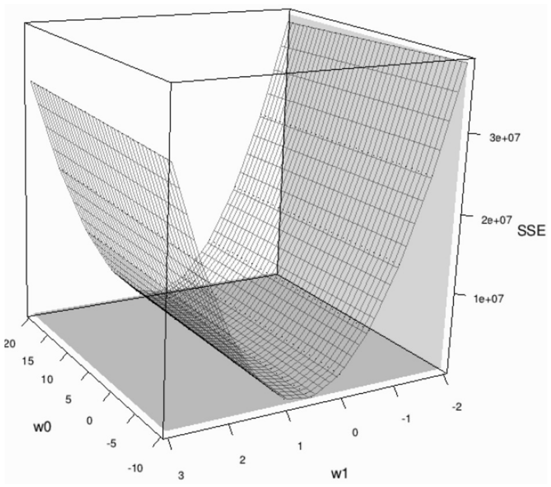
• That is, for a ULR model in which each instance is described with a single descriptive feature, we have the SSE error function L_2 as follows

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}((w_0 + w_1 \times \mathbf{x}_{i1}) - y_i)^2 \quad \textbf{(7.5)}$$

• Different potential models can be combined to form an **error surface** (ES) across which we can search for the **optimal weights** with the minimum SSE.

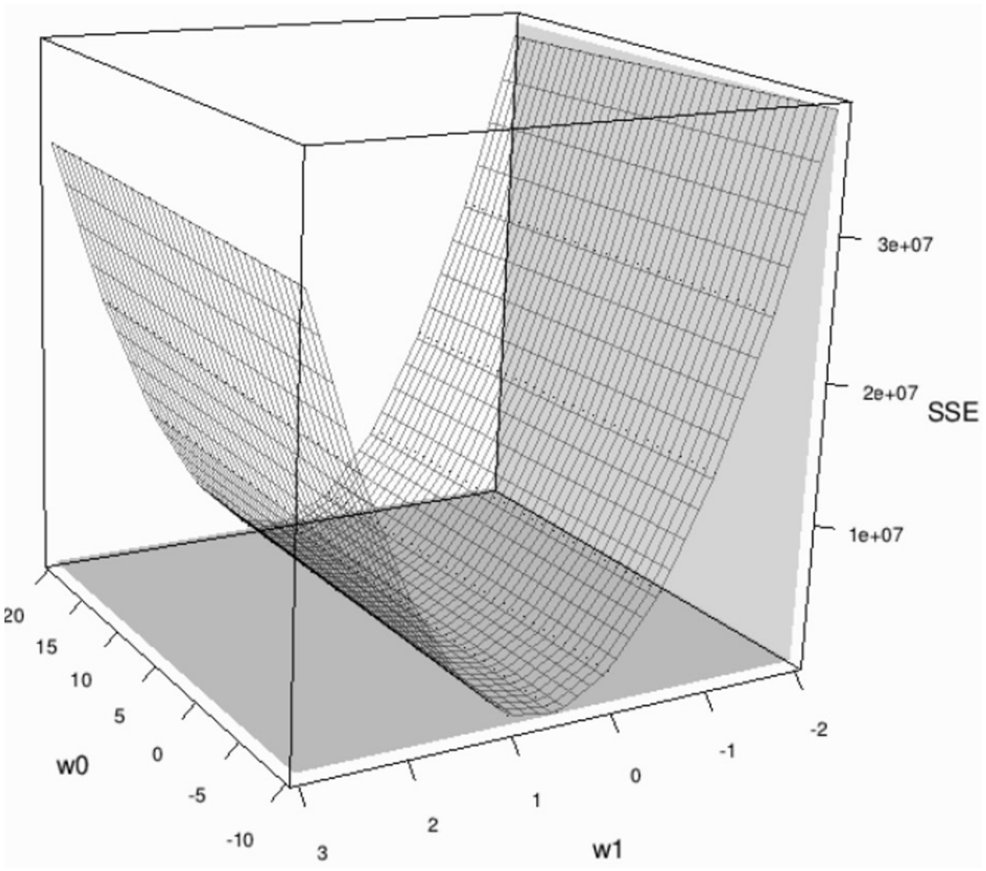
Error Surface (ES)

- The *x*-*y* plane defined by weights w_0 and w_1 is known as a **weight space**.
- weights w_0, w_1, and corresponding SSE make an **error surface** (ES), as shown in Fig. 7.3 below.
- The model that best fits *D* is the model corresponding to the lowest point on the ES.



Error Surface (ES)

- (w_0, w_1, SSE) makes an ES, as shown in Fig. 7.3(a) below



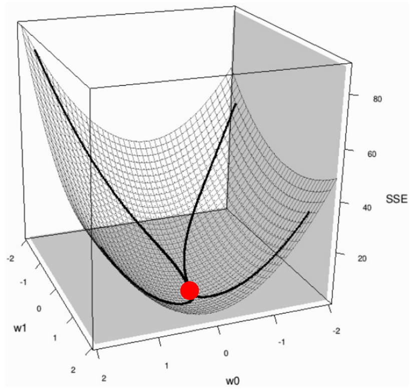
Error Surface (ES)

• We need an efficient way to find the **optimal weights** (i.e., the $\mathbf{w} = (w_0, w_1, \ldots, w_d)$, $d = 1$, with a minimum SSE).

• The ES has two properties

- it is **convex**, (its shape looks like a bowl), and

- it has a **global minimum** (GM). That is, on the ES, there is a unique set of **optimal weights** with the lowest SSE.



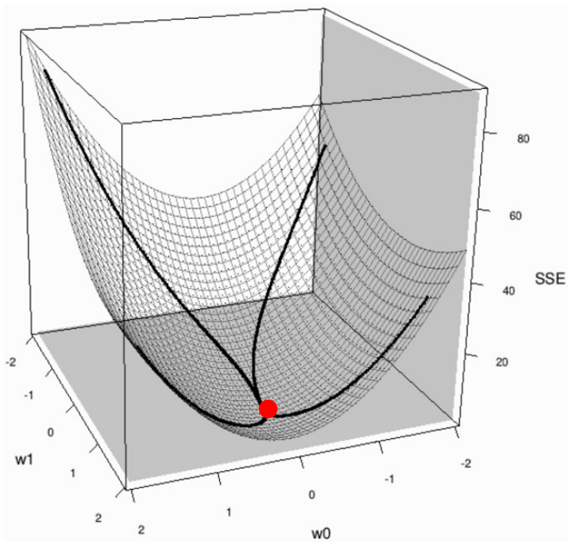
Partial Derivatives (PD)

• If we can find the **global minimum** (GM) of the ES, we can find the set of weights $\mathbf{w} = (w_0, w_1, \ldots, w_d)$, $d = 1$, defining the model $h_{\mathbf{w}}(\mathbf{x})$ that best fits D. This approach to finding weights is known as **least squares optimization**.

• We can find the **optimal weights** at the point where the **partial derivatives** (PD) of the ES w.r.t. w_0 and w_1 are equal to 0 (i.e., $\partial L_2/\partial w_0$ and $\partial L_2/\partial w_1$).

Partial Derivatives (PD)

- The PD of the ES w.r.t. w_0 and w_1 measure the **slope** (or **gradient**) of the ES at the point (w_0, w_1) in the weight space.
- The point *B* on the ES at which the PD w.r.t. w_0 and w_1 are equal to 0 is the point at the very bottom of the bowl defined by the ES.



Partial Derivatives (PD)

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}((w_0 + w_1 \times \mathbf{x}_{i1}) - y_i)^2 \quad \textbf{(7.5)}$$

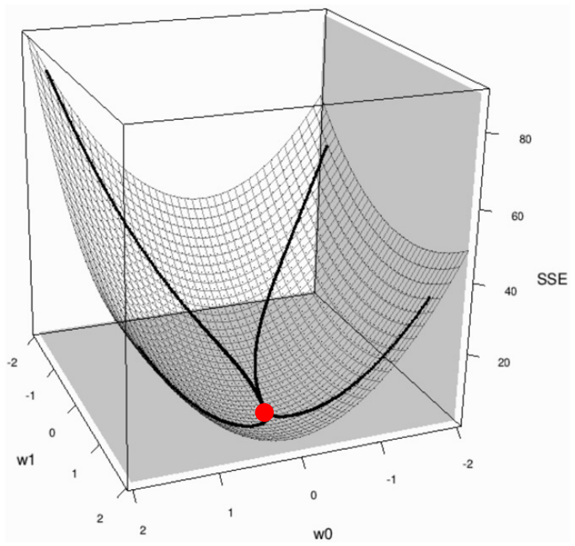
• The point B is at the GM of the ES and the coordinates (w_0, w_1, SSE) of the point B define the **optimal weights** w_0 and w_1 for the prediction model $h_{\mathbf{w}}(\mathbf{x})$ with the lowest SSE on the dataset D.

/* Recall:

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 \quad \textbf{(7.4)}$$

$$h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$$

*/



Partial Derivatives (PD)

- Using the SSE error function L_2,

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}((w_0 + w_1 \times \mathbf{x}_{i1}) - y_i)^2 \quad \textbf{(7.5)}$$

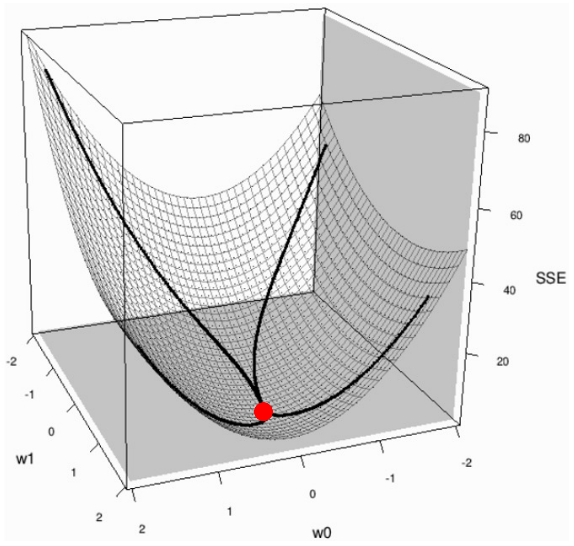
we can define the point $B = (w_0, w_1, \text{SSE})$ on the ES as the point at which

$$\frac{\partial}{\partial w_0} L_2(h_{\mathbf{w}}, D) = \frac{\partial}{\partial w_0}\frac{1}{2}\sum_{i=1}^{m}((w_0 + w_1 \times \mathbf{x}_{i1}) - y_i)^2 = 0 \; \textbf{(7.6)}$$

$$\frac{\partial}{\partial w_1} L_2(h_{\mathbf{w}}, D) = \frac{\partial}{\partial w_1}\frac{1}{2}\sum_{i=1}^{m}((w_0 + w_1 \times \mathbf{x}_{i1}) - y_i)^2 = 0 \; \textbf{(7.7)}$$

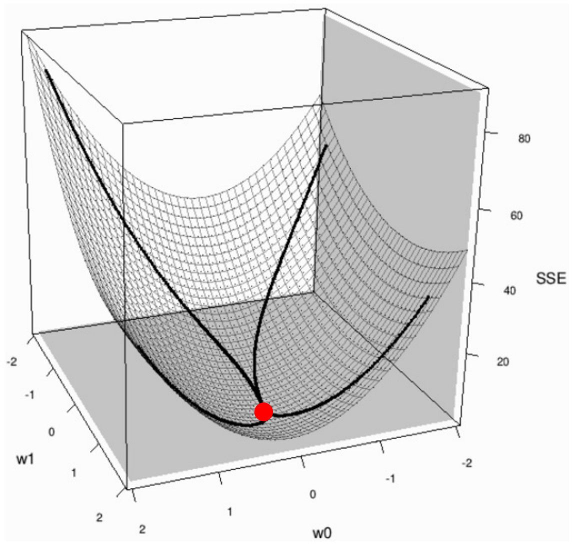
Gradient Descent Algorithm (GDA)

- There are a number of different ways to find the point B = (w_0, w_1, SSE) with the **optimal weights** w_0 and w_1 and the lowest SSE for a given dataset D (e.g., solving **normal equations**, gradient descent algorithm (GDA)).



Gradient Descent Algorithm (GDA)

- We will use the **gradient descent algorithm** (GDA) to find the point $B = (w_0, w_1, \text{SSE})$.
- The GDA is one of the most important algorithms in machine learning.
- We will present the GDA for a **multivariable linear regression** (MLR) model.



Summary

- ULR: $L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ **(7.4)**

where $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$, $\mathbf{x}_i = (1, x_{i1}, x_{i2}, ..., x_{id})$, $\mathbf{w} = (w_0, w_1, ..., w_d)$

- When $d = 1$, we have $\mathbf{x}_i = (1, x_{i1})$, $\mathbf{w} = (w_0, w_1)$

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}((w_0 + w_1 \times \mathbf{x}_{i1}) - y_i)^2 \quad \textbf{(7.5)}$$

Contents

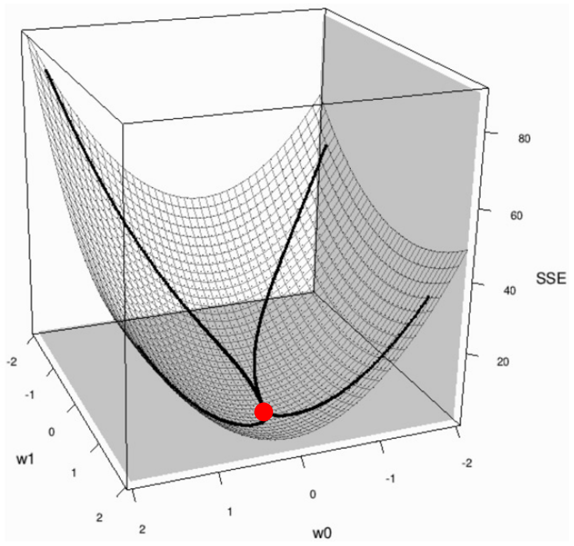
2. Multivariate Linear Regression (MLR)

- The most common approach to error-based machine learning for predictive analytics is to use the MLR with **gradient descent** (GD) to train a best-fit model $h_{\mathbf{w}}$ for a given D.
- A ULR model handles only a single attribute (i.e., $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{id})$, $d = 1$).
- An MLR model handles multiple attributes (i.e., $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{id})$).

/* $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$ */

2. Multivariate Linear Regression (MLR)

- The point B = (w_0, w_1, ..., w_d, SSE) (with the **optimal weights** w_0, w_1, ..., w_d and the lowest SSE for a given dataset D) can be found by the GDA.



2. Multivariate Linear Regression (MLR)

- For one instance **x** in D, we can define an MLR model as

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1}^{d} w_j x_j \quad \textbf{(7.8)}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ is a vector of d descriptive features, and $w_0, w_1, w_2, \ldots, w_d$ are $(m + 1)$ weights (i.e., $\mathbf{w} = (w_0, w_1, w_2, \ldots, w_d)$).

/* $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$ */

2. Multivariate Linear Regression (MLR)

- By adding a dummy descriptive feature $x_0 = 1$. The MLR model becomes

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{j=0}^{d} w_j x_j = \mathbf{w} \cdot \mathbf{x} \quad \mathbf{(7.9)}$$

where $\mathbf{w} \cdot \mathbf{x}$ (or $\langle \mathbf{w}, \mathbf{x} \rangle$) is the **dot product** of the weight vector $\mathbf{w} = (w_0, w_1, w_2, ..., w_d)$ and the feature vector $\mathbf{x} = (1, x_1, x_2, ..., x_d)$.

/* $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$ */

Error Function

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 \quad \textbf{(7.4)}$$

- The SSE **error function** L_2 for the MLR is defined as

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \textbf{(7.10)}$$

where D has m training instances ($\mathbf{x}_i$, y_i), $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$ is the prediction made by a model $h_{\mathbf{w}}$ for a training instance $\mathbf{x}_i = (1, x_{i1}, x_{i2}, ..., x_{id})$, and the model $h_{\mathbf{w}}(\mathbf{x}_i)$ is defined by the **weight vector** $\mathbf{w} = (w_0, w_1, w_2, ..., w_d)$.

Example

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{j=0}^{d} w_j x_j = \mathbf{w} \cdot \mathbf{x} \text{ (7.9)}$$

- Reconsider the problem of predicting office rental prices. The resulting equation of the MLR model is

RENTAL PRICE = $w_0 + w_1$×SIZE + w_2×FLOOR + w_3×BROADBAND RATE

(i.e., $\mathbf{w} = (w_0, w_1, w_2, w_3)$ and $\mathbf{x}$ = (1, SIZE, FLOOR, BROADBAND RATE))

- We will describe how to find the **optimal weights** using the GDA later, for now we set $w_0 = -0.1513$, $w_1 = 0.6270$, $w_2 = -0.1781$, and $w_3 = 0.0714$.

Example

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{j=0}^{d} w_j x_j = \mathbf{w} \cdot \mathbf{x} \ (7.9)$$

- This means that the resulting MLR model is rewritten as

RENTAL PRICE = –0.1513 + 0.6270×SIZE – 0.1781×FLOOR + 0.0714×BROADBAND RATE

(w_0 = –0.1513, w_1 = 0.6270, w_2 = –0.1781, and w_3 = 0.0714)

- Using this model, we can predict the expected rental price of a 690 square foot office on the 11th floor of a building with a broadband rate of 50 Mbps as follows.

Example

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{j=0}^{d} w_j x_j = \mathbf{w} \cdot \mathbf{x} \ (7.9)$$

RENTAL PRICE = –0.1513 + 0.6270×690 – 0.1781×11 + 0.0714×50 = 434.0896

(w_0 = –0.1513, w_1 = 0.6270, w_2 = –0.1781, w_3 = 0.0714, SIZE = 690, FLOOR = 11, BROADBAND RATE = 50)

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

Gradient Descent (GD)

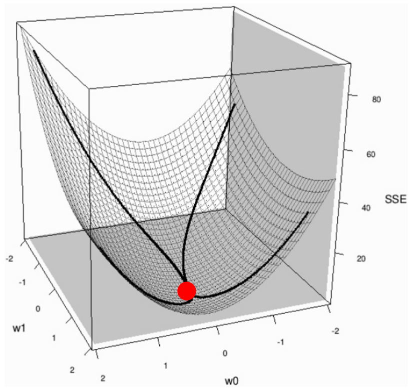
$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

• The GM can be found at the point at which the PD of the ES w.r.t. the weights $\mathbf{w} = (w_0, w_1, w_2, \ldots, w_d)$ are equal to zero (i.e., $\partial L_2/\partial w_j = 0$).

• The **convex** ES has a **global minimum** (GM).

• The approach using a guided search from a random starting position to find the **optimal weights** is known as **gradient descent** (GD).

/* MLR: $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$ */



Gradient Descent (GD)

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{j=0}^{d} w_j x_j = \mathbf{w} \cdot \mathbf{x} \quad \textbf{(7.9)}$$

• The GD approach works as follows.

- selects a random point within the weight space. That is, each weight w_j in the MLR equation

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w} \cdot \mathbf{x}_i - y_i)^2 \quad \textbf{(7.10)}$$

is assigned a random value within some range, and

- calculates the SSE associated with this point based on predictions made for each instance $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{id})$ in D by using the randomly selected weights.

Gradient Descent (GD)

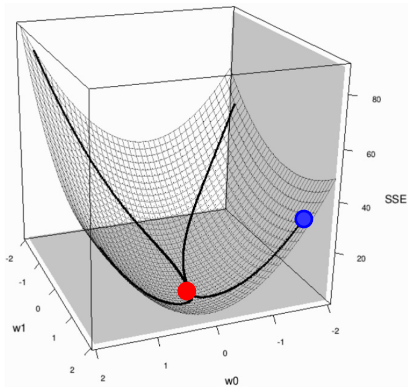
$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \textbf{(7.10)}$$

- It is possible to determine the **slope** (or **gradient**) of the ES by determining the derivative of the function used to generate it (i.e., the SSE loss function L_2), and then calculating the value of this derivative at the random point selected in the weight space.
- This means that the GDA can use the direction of the **slope** of the ES at the current location in the weight space.

Gradient Descent (GD)

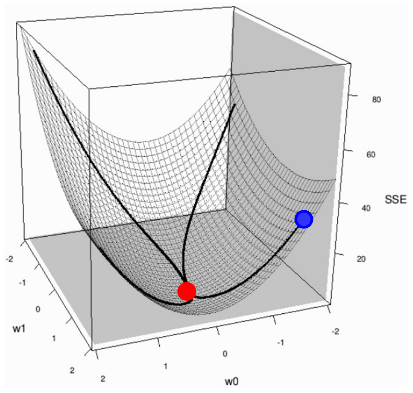
$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \textbf{(7.10)}$$

- The randomly selected weights are adjusted slightly in the direction of the ES **gradient** to move to a new position on the ES.
- Because the adjustments are made in the **direction** of the ES gradient, this new point will be closer to the overall GM.
- This adjustment is repeated over and over until the GM on the ES is reached.



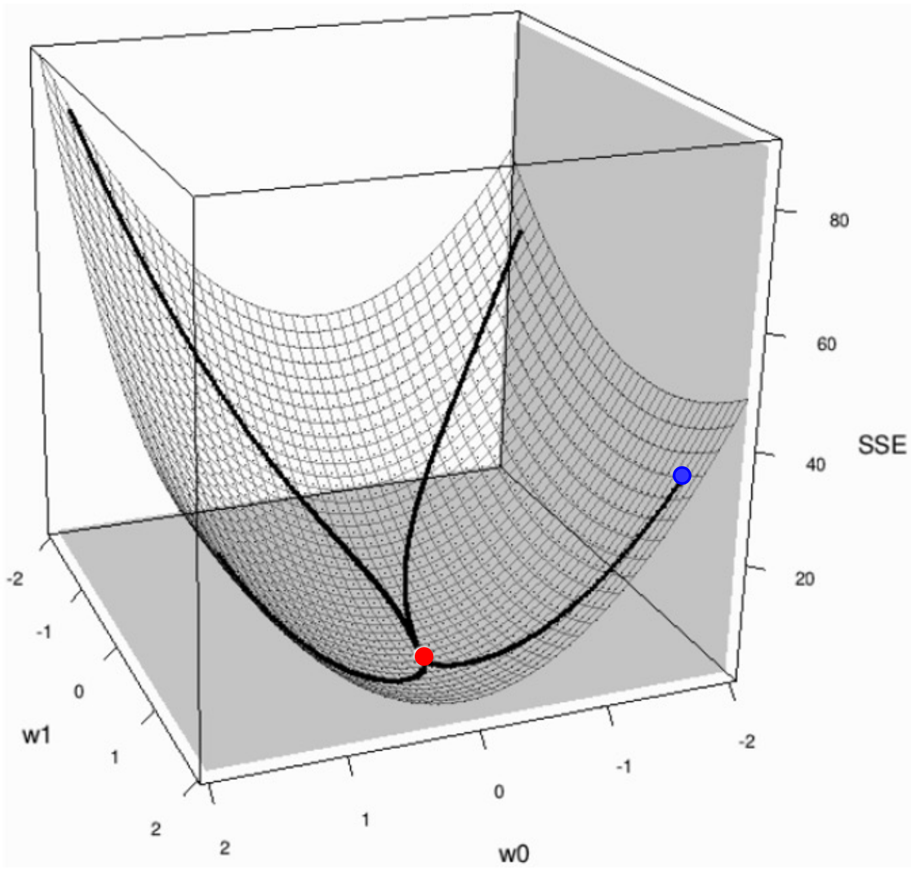
Gradient Descent (GD)

• Fig. 7.4 below shows an ES (defined over just two weights so that we can visualize the ES) and some examples of the path down this surface that the GDA would take from different random starting positions.



Gradient Descent (GD)

- The GM is marked as the red dot in the center.



Gradient Descent Algorithm (GDA)

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \textbf{(7.10)}$$

• The GDA for training MLR models is presented in Algorithm 7.1.

Algorithm 7.1 The GDA for training MLR models.

Inputs: training dataset D, learning rate α, a function *errorDelta*$(D, w_j) = \partial/\partial w_j\, L_2(h_{\mathbf{w}}, D)$ that determines the direction in which to adjust a given weight w_j so as to move down the slope of an ES determined by D, a convergence criterion that indicates that the algorithm has completed.

/* $f(x) = 2x^3, f'(x) = 6x^2, f(x, y) = 2x^3 + 4y^5$,
$\partial/\partial x\, f(x, y) = 6x^2$, $\partial/\partial y\, f(x, y) = 20y^4$ */

Gradient Descent Algorithm (GDA)

```
1. **w** ← random starting point in the weight space
2. **repeat**
3.     **for** each $w_j$ in **w do**
4.         $w_j \leftarrow w_j - \alpha \times errorDelta(D, w_j)$
5: **until** convergence occurs (e.g., error change is negligible)
```

/* $errorDelta(D, w_j) = \partial/\partial w_j \, L_2(h_{\mathbf{w}}, D)$

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

*/

$$\boxed{errorDelta(D, w_j) = \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})}$$

Gradient Descent Algorithm (GDA)

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

• Each weight w_j is iteratively adjusted by a small amount ($w_j \leftarrow w_j - \alpha \times errorDelta(D, w_j)$) based on the error in the predictions made by the current candidate model so as to generate subsequently more and more accurate candidate models.

• Eventually, the algorithm will converge to a point on the ES where any subsequent changes to weights do not lead to a noticeably better model (within some tolerance).

/* $errorDelta(D, w_j) = \partial/\partial w_j \, L_2(h_{\mathbf{w}}, D)$ */

Gradient Descent Algorithm (GDA)

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \textbf{(7.10)}$$

• At this point we can expect the algorithm to have found the GM of the ES and, as a result, the most accurate predictive model possible.

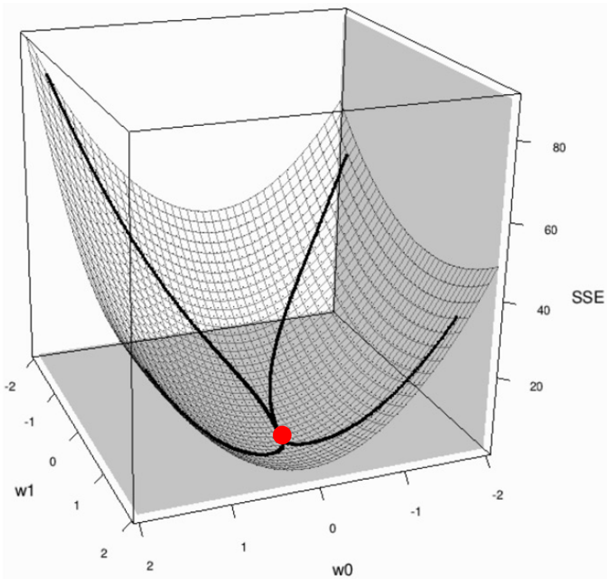
• For each weight w_j, a small adjustment is made by subtracting (or adding) a small value called a **delta value** to the current weight w_j.

/* delta value is computed by

$errorDelta(D, w_j)$

$= \partial/\partial w_j \, L_2(h_{\mathbf{w}}, D)$

*/



Gradient Descent Algorithm (GDA)

$$errorDelta(D, w_j) = \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})$$

• The adjustment should ensure that the change in the weight leads to a move downward on the ES.

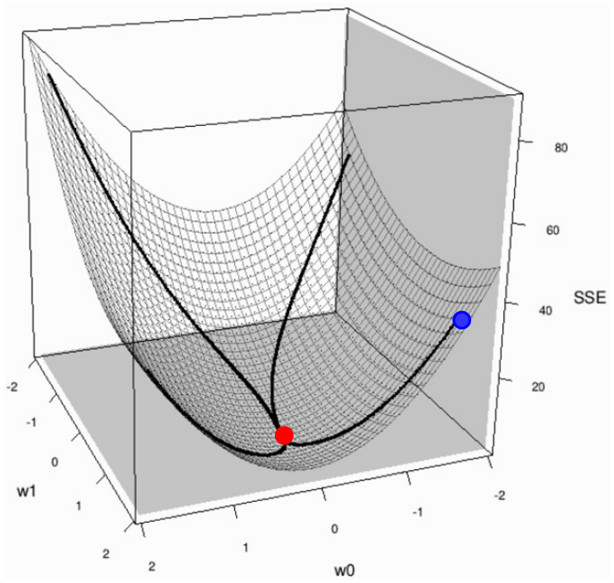
• The **learning rate** α determines the size of the adjustments made to weights at each iteration of the algorithm.

/*

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$$

$$= \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w} \cdot \mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

*/



Error Delta Function

$$errorDelta(D, w_j) = \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m} ((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})$$

- The **error delta function** $errorDelta(D, w_j) = \partial/\partial w_j\ L_2(h_{\mathbf{w}}, D)$: calculates the **delta value** that determines the direction (either positive or negative) and the magnitude of the adjustments made to each weight w_j ($w_j \leftarrow w_j - \alpha \times errorDelta(D, w_j)$)
- The direction and magnitude of the adjustment to be made to a weight w_j is determined by the **gradient** of the ES at the current position in the weight space.

Error Delta Function

- Recall that the ES is defined by the SSE **error function** L_2

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \textbf{(7.10)}$$

- The **gradient** at any point on the ES is given by the value of the PD of the **error function** w.r.t. a particular weight w_j at that point.

/* we want to have

$$errorDelta(D, w_j) = \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij}) \quad */$$

Error Delta Function

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

- The **error delta function** *errorDelta*(D, w_j) = $\partial/\partial w_j\ L_2(h_{\mathbf{w}}, D)$ performs this calculation to determine the **delta value** by which each weight should be adjusted.
- We will show that

$$\boxed{errorDelta(D, w_j) = \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})}$$

/* $(x^\alpha)' = \alpha x^{\alpha-1}$, $(u^\alpha)' = \alpha u^{\alpha-1}u'$

$f(x) = 2x^3, f'(x) = 6x^2, f(x, y) = 2x^3 + 4y^5,$

$\partial/\partial x\ f(x, y) = 6x^2,\ \partial/\partial y\ f(x, y) = 20y^4$ */

Error Delta Function - for one instance x

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

- To understand how to calculate the value of the PD of the **error function** w.r.t. a particular weight w_j, let us imagine for a moment that D contains one instance ($\mathbf{x}$, y).
- For one training instance **x**, the **gradient** of the ES (i.e., **slope** of the error function L_2) is given as the PD of L_2 w.r.t. each weight w_j

/* we want to have

$$errorDelta(D, w_j) = \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij}) \quad */$$

Error Delta Function - for one instance x

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

- $$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \frac{\partial}{\partial w_j}\left(\frac{1}{2}(h_{\mathbf{w}}(\mathbf{x}) - y)^2\right) \quad \mathbf{(7.11)}$$

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (h_{\mathbf{w}}(\mathbf{x}) - y) \times \frac{\partial}{\partial w_j}(\ h_{\mathbf{w}}(\mathbf{x}) - y) \quad \mathbf{(7.12)}$$

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (h_{\mathbf{w}}(\mathbf{x}) - y) \times \frac{\partial}{\partial w_j}((\mathbf{w}\cdot\mathbf{x}) - y) \quad \mathbf{(7.13)}$$

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (h_{\mathbf{w}}(\mathbf{x}) - y) \times x_j \quad \mathbf{(7.14)}$$

$$\boxed{\begin{array}{l} (x^2)' = 2x \\ (u^2)' = 2u \times u' \\ f(x, y) = ax + by \\ \partial f/\partial x = a \\ \partial f/\partial y = b \end{array}}$$

(x_j is the jth component of $\mathbf{x}$)

/* $\mathbf{w}\cdot\mathbf{x} = w_0x_0 + \ldots + w_jx_j + \ldots + w_dx_d$ */

Error Delta Function - for *m* instances $\mathbf{x}_i$

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

- For *m* training instances, we calculate the SSE for each instance $\mathbf{x}_i = (1, x_{i1}, ..., x_{ij}, ..., x_{id})$ so $\partial/\partial w_j$ $L_2(h_{\mathbf{w}}, D) = (h_{\mathbf{w}}(\mathbf{x}) - y) \times x_j$ **(7.14)** becomes

$$\frac{\partial}{\partial w_j}L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij}) \quad \mathbf{(7.15)}$$

where $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)$ are *m* training instances, and x_{ij} is the *j*th feature of instance $(\mathbf{x}_i, y_i)$.

/* $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w}\cdot\mathbf{x}_i = w_0x_{i0} + ... + w_jx_{ij} + ... + w_dx_{id}$ */

Error Delta Function - for *m* instances $\mathbf{x}_i$

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

- The **error delta function** is defined as

$$\boxed{\begin{aligned} errorDelta(D, w_j) &= \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) \\ &= \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i)\times x_{ij}) \quad \mathbf{(7.16)} \end{aligned}}$$

Error Delta Function - for *m* instances $\mathbf{x}_i$

$$w_j \leftarrow w_j - \alpha \times errorDelta(D, w_j)$$

• Line 4 of Algorithm 7.1 can therefore be rewritten as what is known as the **weight update rule** for MLR with **gradient descent** (GD):

$$w_j \leftarrow w_j - \alpha \times \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})$$

where w_j is the *j*th component of **w**, α is a **learning rate**, y_i is the expected target value for $\mathbf{x}_i$, $h_{\mathbf{w}}(\mathbf{x}_i)$ is the prediction made for $\mathbf{x}_i$ by the current candidate model defined by **w**, and x_{ij} is the *j*th feature of $\mathbf{x}_i$ and corresponds with weight w_j in the MLR model.

Gradient Descent Algorithm (GDA)

- Algorithm 7.1 can be rewritten as follows.

Algorithm 7.1 The GDA for training MLR models.

Inputs: training dataset D, learning rate α,
a convergence criterion that indicates that
the algorithm has completed

Gradient Descent Algorithm (GDA)

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

```
1. w ← random starting point in the weight space
2. repeat
3.     for each $w_j$ in w do
4.         $w_j \leftarrow w_j - \alpha \times \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})$
5: until convergence occurs (e.g., $|\mathrm{SSE}_k - \mathrm{SSE}_{k-1}| \leq \varepsilon = 0.001$)
```

/* MLR: $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w}\cdot\mathbf{x}_i$

$$errorDelta(D, w_j) = \frac{\partial}{\partial w_j}L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})$$ */

Summary

- ULR: $L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}((w_0 + w_1 \times \mathbf{x}_{i1}) - y_i)^2$ **(7.5)**
- MLR: $L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ **(7.4)**

where $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$, $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{id})$, $\mathbf{w} = (w_0, w_1, \ldots, w_d)$

$$errorDelta(D, w_j) = \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})$$

$$w_j \leftarrow w_j - \alpha \times errorDelta(D, w_j)$$

Contents

3. Multivariate Logistic Regression (logit)

- An MLR model can be used to predict a continuous target feature.
- A multivariable logistic regression (logit) can be used to predict a categorical target feature.
- Table 7.6 shows a sample dataset with a categorical target feature.

/*

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

MLR: $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$

*/

3. Multivariate Logistic Regression (logit)

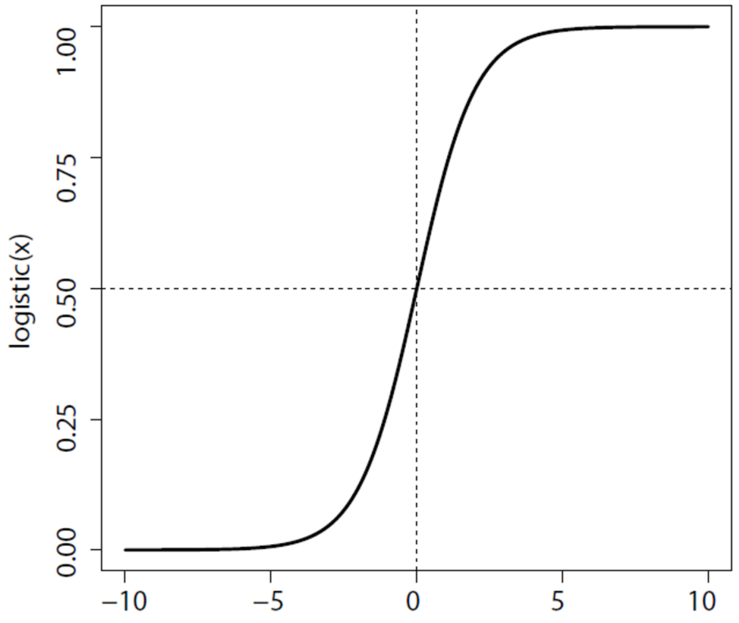
- Table 7.6: A dataset listing features for a number of generators.

ID	RPM	Vibration	Status
1	568	585	good
2	586	565	good
...	...	...	...
28	933	330	good
29	562	309	faulty
30	578	346	faulty
...	...	...	...
56	939	99	faulty

(RPM: revolutions per minute)

Logistic Function (LF)

- The LF is given by $g(x) = 1 / (1 + e^{-x}) \in (0, 1)$, where x is a numeric value and $e = 2.7183$.
- A plot of the LF for values of x in the range [–10, 10] is shown in Fig. 7.12 below.



3. Multivariate Logistic Regression (logit)

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

• We can define a multivariable logistic regression (logit) model as

$$h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w}\cdot\mathbf{x}) = 1 / (1 + e^{-\mathbf{w}\cdot\mathbf{x}}) \in (0, 1) \ \mathbf{(7.26)}$$

(recall: an MLR model is $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}$)

• To find the optimal DB for a logit problem, we use the GDA (Algorithm 7.1) to minimize the SSE based on the training dataset *D*.

/*

MLR: $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w}\cdot\mathbf{x}_i$

*/

logit - for one instance x

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

- The **error delta function** *errorDelta*(D, w_j) = $\partial/\partial w_j\ L_2(h_{\mathbf{w}}, D)$ used in the **weight update rule** given on Line 4 of the GDA (Algorithm 7.1)

$$w_j \leftarrow w_j - \alpha \times \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})$$

is modified to train a logit model, where $h_{\mathbf{w}}(\mathbf{x}_i) = g(\mathbf{w}\cdot\mathbf{x}_i)$.

- For one training instance ($\mathbf{x}$, y), the new **weight update rule** is derived by computing the PD of the error function L_2, $\partial/\partial w_j\ L_2(h_{\mathbf{w}}, D)$.

Error Delta Function *errorDelta*(D, w_j)

$$L_2(h_\mathbf{w}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_\mathbf{w}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

- The PD of the error function L_2 is

$$\frac{\partial}{\partial w_j} L_2(h_\mathbf{w}, D) = \frac{\partial}{\partial w_j}\frac{1}{2}(h_\mathbf{w}(\mathbf{x}) - y)^2$$

where w_j is the jth component of

$\mathbf{w} = (w_0, w_1, \ldots, w_d)$.

$(x^2)' = 2x$
$(u^2)' = 2u \times u'$
$f(x, y) = ax + by$
$\partial f/\partial x = a$
$\partial f/\partial y = b$

- Applying the chain rule to this, we get

$$\frac{\partial}{\partial w_j} L_2(h_\mathbf{w}, D) = (h_\mathbf{w}(\mathbf{x}) - y) \times \frac{\partial}{\partial w_j}(h_\mathbf{w}(\mathbf{x}) - y)$$

/* $\partial/\partial w_j\, L_2(h_\mathbf{w}, D) = (h_\mathbf{w}(\mathbf{x}) - y) \times x_j$ **(7.14)** */

Error Delta Function *errorDelta*(D, w_j)

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \mathbf{(7.10)}$$

- But $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x})$, so we have

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (g(\mathbf{w}\cdot\mathbf{x}) - y) \times \frac{\partial}{\partial w_j}(g(\mathbf{w}\cdot\mathbf{x}) - y)$$

- Applying the chain rule again to the PD part of this equation, and remembering that $\partial/\partial w_j\ (\mathbf{w} \cdot \mathbf{x}) = x_j$, we get

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (g(\mathbf{w}\cdot\mathbf{x}) - y) \times \frac{\partial}{\partial w_j} g(\mathbf{w}\cdot\mathbf{x}) \times \frac{\partial}{\partial w_j}\mathbf{w}\cdot\mathbf{x}$$

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (g(\mathbf{w}\cdot\mathbf{x}) - y) \times \frac{\partial}{\partial w_j} g(\mathbf{w}\cdot\mathbf{x}) \times x_j$$

Error Delta Function *errorDelta*(D, w_j)

$$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{m}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2 \quad \textbf{(7.10)}$$

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (g(\mathbf{w}\cdot\mathbf{x}) - y) \times \frac{\partial}{\partial w_j} g(\mathbf{w}\cdot\mathbf{x}) \times x_j$$

- The derivative of the LF $g(x)$ is

$$d/dx\, g(x) = g(x)(1 - g(x)).$$

- Thus, we obtain

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (g(\mathbf{w}\cdot\mathbf{x}) - y) \times g(\mathbf{w}\cdot\mathbf{x}) \times (1 - g(\mathbf{w}\cdot\mathbf{x})) \times x_j \quad \textbf{(7.29)}$$

Error Delta Function *errorDelta(D, w_j)*

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (g(\mathbf{w} \cdot \mathbf{x}) - y) \times g(\mathbf{w} \cdot \mathbf{x}) \times (1 - g(\mathbf{w} \cdot \mathbf{x})) \times x_j$$

- Rewriting $g(\mathbf{w} \cdot \mathbf{x})$ as $h_{\mathbf{w}}(\mathbf{x})$, we get

$$\frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = (h_{\mathbf{w}}(\mathbf{x}) - y) \times h_{\mathbf{w}}(\mathbf{x}) \times (1 - h_{\mathbf{w}}(\mathbf{x})) \times x_j$$

$$\rightarrow errorDelta(D, w_j) = (h_{\mathbf{w}}(\mathbf{x}) - y) \times h_{\mathbf{w}}(\mathbf{x}) \times (1 - h_{\mathbf{w}}(\mathbf{x})) \times x_j$$

- This is the PD of the ES w.r.t. a particular weight w_j and indicates the gradient of the ES.

/* MLR: $\partial/\partial w_j\ L_2(h_{\mathbf{w}}, D) = (h_{\mathbf{w}}(\mathbf{x}) - y) \times x_j$ **(7.14)** */

Weight Update Rule - for one instance x

• Using the formulation for the gradient of the ES (i.e., *errorDelta*(D, w_j) = $\partial/\partial w_j\, L_2(h_{\mathbf{w}}, D)$), we can write the **weight update rule** for logit as

$$w_j \leftarrow w_j - \alpha \times (h_{\mathbf{w}}(\mathbf{x}) - y) \times h_{\mathbf{w}}(\mathbf{x}) \times (1 - h_{\mathbf{w}}(\mathbf{x})) \times x_j,$$

where $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = 1 / (1 + e^{-\mathbf{w}\cdot\mathbf{x}})$.

/* MLR: $\partial/\partial w_j\, L_2(h_{\mathbf{w}}, D) = (h_{\mathbf{w}}(\mathbf{x}) - y) \times x_j$ **(7.14)** */

Weight Update Rule - for *m* instances $\mathbf{x}_i$

- For *m* training instances ($\mathbf{x}_i$, y_i), the **weight update rule** for logit is

$$w_j \leftarrow w_j - \alpha \times \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times h_{\mathbf{w}}(\mathbf{x}_i) \times (1 - h_{\mathbf{w}}(\mathbf{x}_i)) \times x_{ij})$$

i.e., we have

$$errorDelta(D, w_j) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times h_{\mathbf{w}}(\mathbf{x}_i) \times (1 - h_{\mathbf{w}}(\mathbf{x}_i)) \times x_{ij})$$

/* MLR:

$$errorDelta(D, w_j) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij}) \quad \textbf{(7.16)}$$

*/

Decision Boundary (DB) for logit

- Recall: A logit model is

$$h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = 1 / (1 + e^{-\mathbf{w}\cdot\mathbf{x}}) \in (0, 1) \quad \textbf{(7.26)}$$

- The found optimal DB for a logit model can be drawn by plotting the graph of $\mathbf{w} \cdot \mathbf{x} = 0$.
- Example

$\mathbf{w} = (w_0, w_1, w_2)$, $\mathbf{x} = (1, x_1, x_2)$

$\mathbf{w} \cdot \mathbf{x} = 0 \Leftrightarrow w_0 + w_1x_1 + w_2x_2 = 0$

$\Leftrightarrow x_2 = (-w_0 - w_1x_1) / w_2$

$\Leftrightarrow x_2 = -(w_1 / w_2)x_1 - (w_0 / w_2)$ // $y = ax + b$

Summary

- ULR: $L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}((w_0 + w_1 \times \mathbf{x}_{i1}) - y_i)^2$ **(7.5)**

- MLR: $L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ **(7.4)**

where $h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i$.

$$errorDelta(D, w_j) = \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})$$

- logit: $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = 1 \,/\, (1 + e^{-\mathbf{w} \cdot \mathbf{x}}) \in (0, 1)$

$$errorDelta(D, w_j) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times h_{\mathbf{w}}(\mathbf{x}_i) \times (1 - h_{\mathbf{w}}(\mathbf{x}_i)) \times x_{ij})$$

$$w_j \leftarrow w_j - \alpha \times errorDelta(D, w_j)$$

Contents

4. Polynomial Regression (PR)

- The MLR and logit models that we have looked at so far model a **linear relationship** between descriptive features **x** and a target feature *y*.
- Linear models work very well when the underlying relationships in the data are linear.
- If the underlying data exhibits a **non-linear relationship**, we need to use a **polynomial regression** (PR) model.

4. Polynomial Regression (PR)

- For example, the dataset in Table 7.9 is based on an agricultural scenario and shows rainfall RAIN (in mm per day) and resulting grass growth GROWTH (in kilograms per acre per day), measured on a number of Irish farms during July 2012.

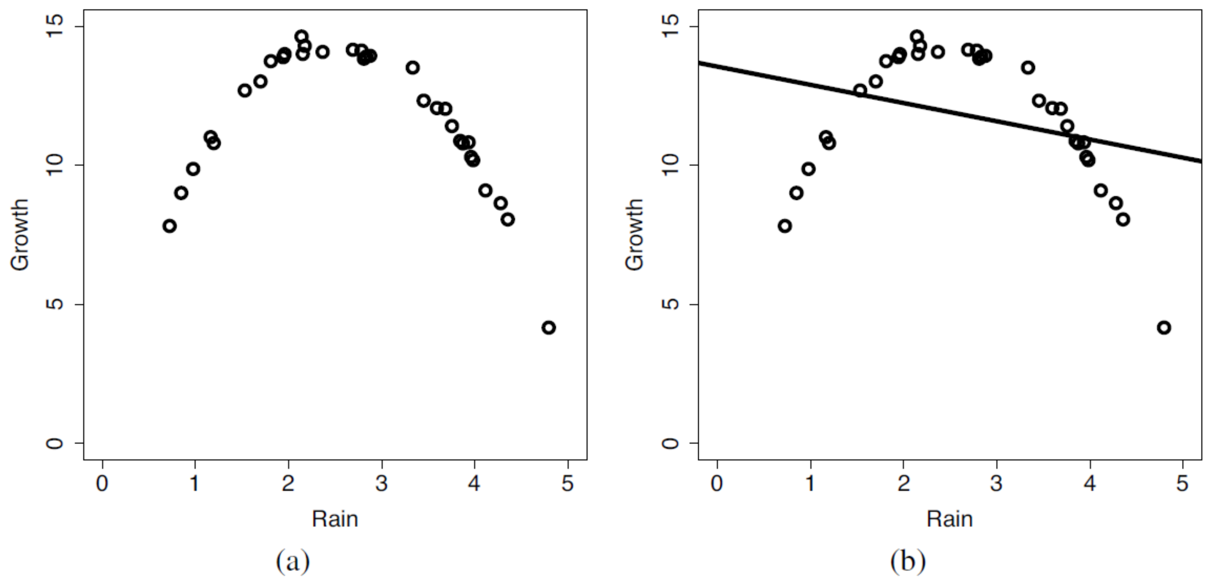
4. Polynomial Regression (PR)

- Table 7.9: A dataset describing grass growth on Irish farms during July 2012.

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

4. Polynomial Regression (PR)

• Fig. 7.16: (a) A scatter plot of the RAIN and GROWTH feature from the grass growth dataset; (b) the same plot with a ULR model trained to capture the **non-linear relationship** between the RAIN and GROWTH.



4. Polynomial Regression (PR)

- To successfully model the **non-linear relationship** between rainfall RAIN and grass growth GROWTH, we need to introduce non-linear elements.
- A generalized way to capture the **non-linear relationship** is to introduce **basis functions** that transform the raw inputs to the model into non-linear representations but still keep the model linear in terms of the weights.

4. Polynomial Regression (PR)

- The advantage of using **basis functions** is that we do not need to make any other changes to the approach we have presented so far.
- **Basis functions** work for both MLR models that predict a continuous target feature and logit models that predict a categorical target feature.

Use Basis Functions for MLR

- To use basis functions for the MLR model, we recast the equation

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} = \sum_{j=0}^{d} w_j x_j$$

as follows

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{k=0}^{b} w_k \times \phi_k(\mathbf{x}) \quad \textbf{(7.34)}$$

where $\mathbf{x} = (1, x_1, x_2, \ldots, x_d)$, $\mathbf{w} = (w_0, w_1, \ldots, w_b)$, and $\phi_0, \ldots, \phi_b$ are $b + 1$ **basis functions** that each transform the input vector $\mathbf{x}$ in a different way.

Use Basis Functions for MLR

- Usually, b is quite a bit larger than d (i.e., there are usually more **basis functions** than there are descriptive features).
- One of the most common uses of **basis functions** in MLR is to train models to capture **polynomial relationships**.
- The most common form of **polynomial relationship** is the **second order polynomial** (a.k.a. the **quadratic function**), which takes the general form $c = bx + ax^2$.

Use Basis Functions for MLR

$$h_{\mathbf{w}}(\mathbf{x}) = w_0\phi_0 + \ldots + w_b\phi_b \quad \mathbf{(7.34)}$$

• The relationship between rainfall RAIN and grass growth GROWTH in the grass growth dataset D can be accurately represented as a **second order polynomial** through the following model

$$\text{GROWTH} = w_0 \times \phi_0(\text{RAIN}) + w_1 \times \phi_1(\text{RAIN}) + w_2 \times \phi_2(\text{RAIN}),$$

where $\phi_0(\text{RAIN}) = 1$, $\phi_1(\text{RAIN}) = \text{RAIN}$, $\phi_2(\text{RAIN}) = \text{RAIN}^2$.

• That is, we have $h_{\mathbf{w}}(\mathbf{x}) = \text{GROWTH} = w_0 + w_1 \times \text{RAIN} + w_2 \times \text{RAIN}^2$, where $\mathbf{x} = (1, \text{RAIN})$.

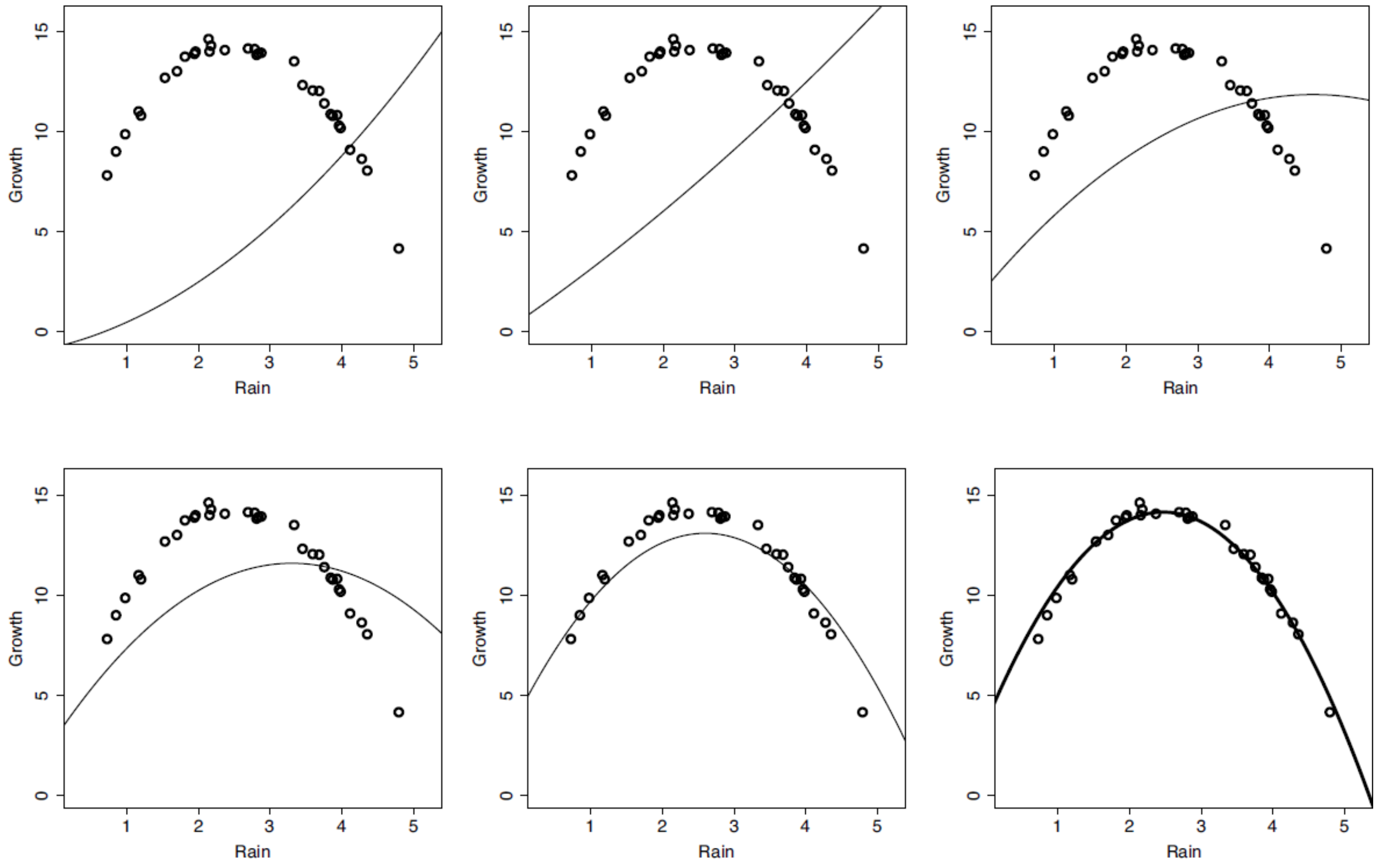
Use Basis Functions for MLR

• What makes this approach really attractive is that, although this new model stated in terms of **basis functions** captures the non-linear relationship between rainfall RAIN and grass growth GROWTH, the model $h_{\mathbf{w}}(\mathbf{x})$ is still linear in terms of the weights and so it can be trained using the GDA without making any changes.

Use Basis Functions for MLR

- Fig. 7.17 below shows the final non-linear model that results from the GD training process, along with a number of the interim steps on the way to this model.

Use Basis Functions for MLR



Use Basis Functions for MLR

$h_{\mathbf{w}}(\mathbf{x}) = w_0\phi_0 + \ldots + w_b\phi_b$ **(7.34)**

• The final **polynomial regression** model is GROWTH = $3.707 \times \phi_0(\text{RAIN}) + 8.475 \times \phi_1(\text{RAIN}) + (-1.717) \times \phi_2(\text{RAIN})$ where $\phi_0(\text{RAIN}) = 1$, $\phi_1(\text{RAIN}) = \text{RAIN}$, $\phi_2(\text{RAIN}) = \text{RAIN}^2$, $\mathbf{w} = (w_0, w_1, w_2) = (3.707, 8.475, -1.717)$. (i.e., GROWTH = $3.707 + 8.475 \times \text{RAIN} + (-1.717) \times \text{RAIN}^2$)

• The above model captures the **nonlinear relationship** in the dataset *D* very well but was still easy to train using the GDA.

Use Basis Functions for logit

- **Basis functions** can also be used to train logit models for categorical prediction problems that involve **nonlinear relationship**.
- Table 7.10 shows the EEG (ElectroEncephaloGraphy) dataset *D*.

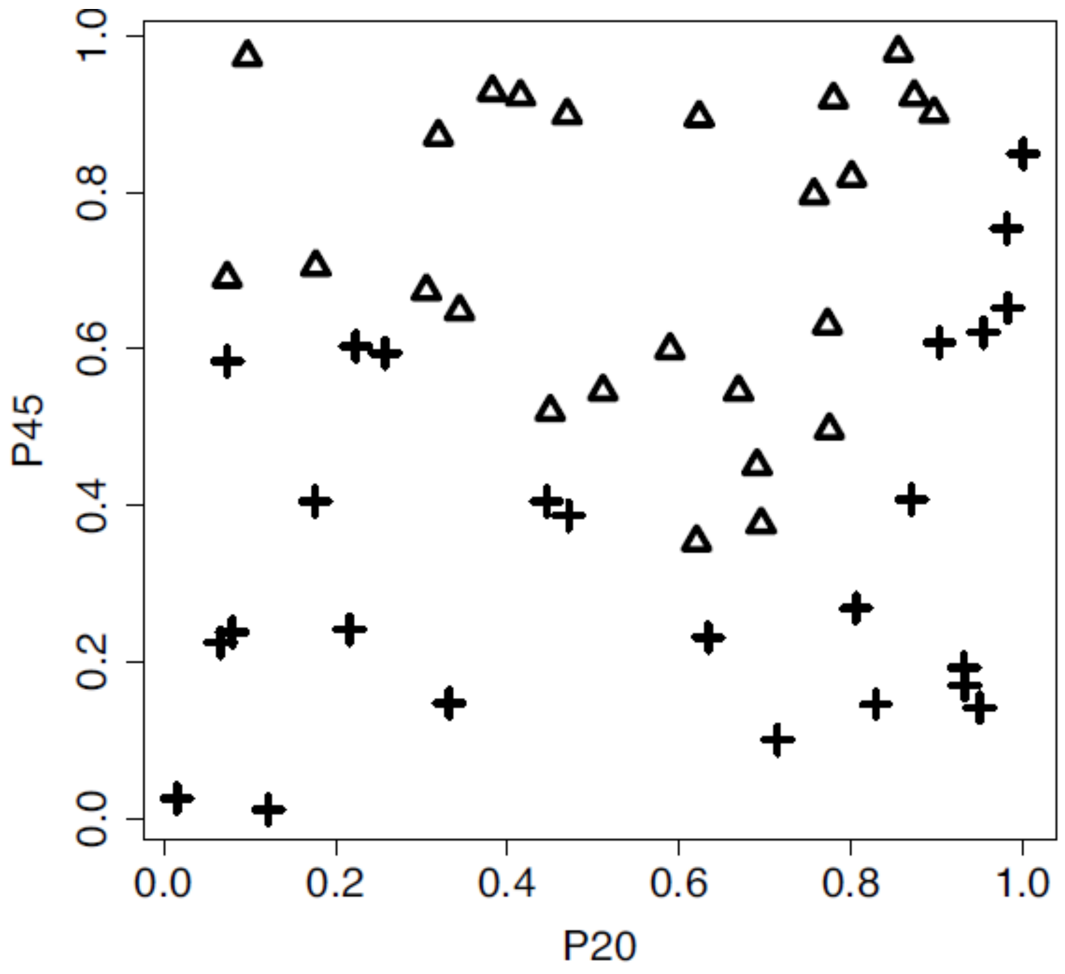
ID	P20	P45	TYPE	ID	P20	P45	TYPE
1	0.4497	0.4499	negative	26	0.0656	0.2244	positive
2	0.8964	0.9006	negative	27	0.6336	0.2312	positive
3	0.6952	0.3760	negative	28	0.4453	0.4052	positive
4	0.1769	0.7050	negative	29	0.9998	0.8493	positive
5	0.6904	0.4505	negative	30	0.9027	0.6080	positive
6	0.7794	0.9190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

Use Basis Functions for logit

- Two descriptive features are P20 and P45, and the target feature is TYPE indicating whether the subject was viewing a positive or a negative image.
- Fig. 7.18 below shows a scatter plot of the dataset *D*, from which it is clear that the DB between the two different types of images is not linear (i.e., the two types of images are **not linearly separable**).

Use Basis Functions for logit

•



Use Basis Functions for logit

• The **non-linear DB** in the above figure can be represented using a **third-order polynomial** in the two descriptive features P20 and P45 (i.e., $\mathbf{x}$ = (1, P20, P45)).

• The MLR model cannot cope with a **non-linear DB** like the one seen in the above figure.

• We can rewrite the logit equation

$$h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = 1 / (1 + e^{-\mathbf{w}\cdot\mathbf{x}}) \in (0, 1)$$

to use **basis functions** as follows.

Use Basis Functions for logit

$$h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = 1 / (1 + e^{-\mathbf{w}\cdot\mathbf{x}}) \in (0, 1)\ \mathbf{(7.26)}$$

$$h_{\mathbf{w}}(\mathbf{x}) = 1 / [1 + \exp(-\sum_{j=0}^{b} w_j \times \phi_j(\mathbf{x}))] \in (0, 1),$$

where $\mathbf{w} = (w_0, w_1, \ldots, w_b)$, $\mathbf{x} = (1, x_1, x_2, \ldots, x_d)$.

• Using the above representation with the following set of **basis functions** will give the learning process the flexibility to find the **non-linear DB** required to successfully separate the different types of images in the EEG dataset *D*.

Use Basis Functions for logit

- The **basis functions** ϕ_0, ..., ϕ_b are defined as follows.

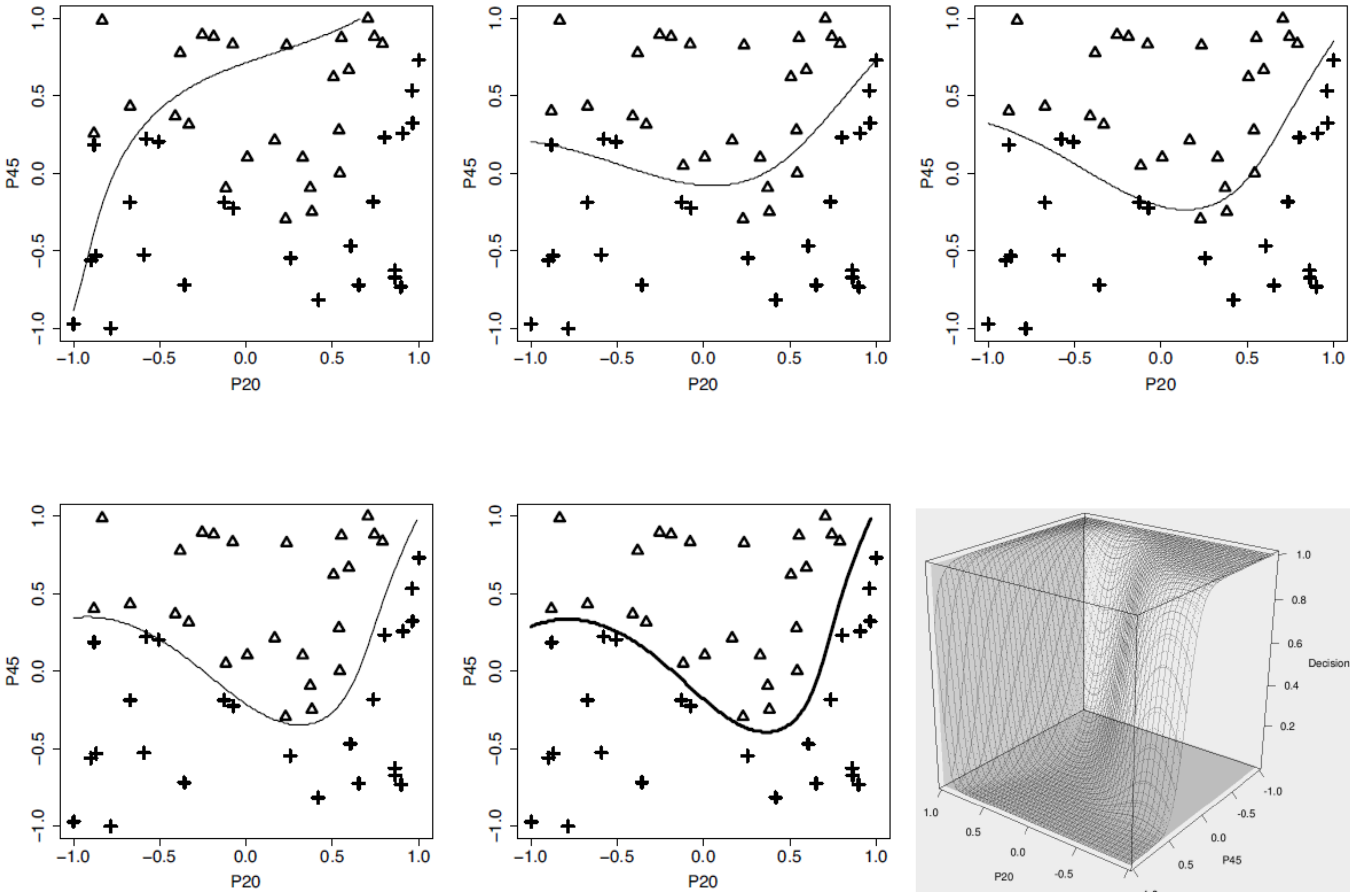
$\phi_0(\text{P20}, \text{P45}) = 1$	$\phi_4(\text{P20}, \text{P45}) = \text{P45}^2$
$\phi_1(\text{P20}, \text{P45}) = \text{P20}$	$\phi_5(\text{P20}, \text{P45}) = \text{P20}^3$
$\phi_2(\text{P20}, \text{P45}) = \text{P45}$	$\phi_6(\text{P20}, \text{P45}) = \text{P45}^3$
$\phi_3(\text{P20}, \text{P45}) = \text{P20}^2$	$\phi_7(\text{P20}, \text{P45}) = \text{P20} \times \text{P45}$

(That is, we have $\mathbf{x} = (x_1, x_2) = (\text{P20}, \text{P45})$, $\phi_0(\mathbf{x}) = 1$, $\phi_1(\mathbf{x}) = x_1$, $\phi_2(\mathbf{x}) = x_2$, $\phi_3(\mathbf{x}) = x_1^2$, $\phi_4(\mathbf{x}) = x_2^2$, $\phi_5(\mathbf{x}) = x_1^3$, $\phi_6(\mathbf{x}) = x_2^3$, $\phi_7(\mathbf{x}) = x_1 \times x_2$.)

Use Basis Functions for logit

- The above model can be trained using the GDA to find the optimal **nonlinear DB** between the two different types of images.
- Fig. 7.19 shows a series of the models built during the GD process.

Use Basis Functions for logit



Use Basis Functions for logit

- The final **polynomial regression** model can accurately distinguish between the two different types of image based on the two features P20 and P45.
- It can be seen that basis functions can be used to capture **nonlinear relationship** in linear and logistic regression models.

Disadvantages of using Basis Functions

- The analyst has to design the basis function set that will be used. This can be a considerable challenge.
- As the number of basis functions grows beyond the number of descriptive features, the complexity of our models increases, so the GD process must search through a more complex weight space.

Summary

- MLR: $$L_2(h_{\mathbf{w}}, D) = \frac{1}{2}\sum_{i=1}^{m}(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2 \quad \textbf{(7.4)}$$

$$errorDelta(D, w_j) = \frac{\partial}{\partial w_j} L_2(h_{\mathbf{w}}, D) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times x_{ij})$$

- logit: $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = 1 / (1 + e^{-\mathbf{w}\cdot\mathbf{x}}) \in (0, 1)$

$$errorDelta(D, w_j) = \sum_{i=1}^{m}((h_{\mathbf{w}}(\mathbf{x}_i) - y_i) \times h_{\mathbf{w}}(\mathbf{x}_i) \times (1 - h_{\mathbf{w}}(\mathbf{x}_i)) \times x_{ij})$$

- Basis functions for MLR: $h_{\mathbf{w}}(\mathbf{x}) = \sum_{k=0}^{b} w_k \times \phi_k(\mathbf{x}) \quad \textbf{(7.34)}$

- Basis functions for logit:

$$h_{\mathbf{w}}(\mathbf{x}) = 1 / [1 + \exp(-\sum_{j=0}^{b} w_j \times \phi_j(\mathbf{x}))] \in (0, 1),$$

Contents

5. Multiclass Logistic Regression

- The **multiclass/multinomial logistic regression** model (a.k.a. **maximum entropy**, **conditional maximum entropy**, or **MaxEnt**) is an extension that handles categorical target features with more than two levels (or classes).
- A good way to build a multiclass logistic regression (multiclass logit) model is the use of a set of **one-versus-all** logit models.

/* $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x}) = 1 / (1 + e^{-\mathbf{w}\cdot\mathbf{x}}) \in (0, 1)$ **(7.26)** */

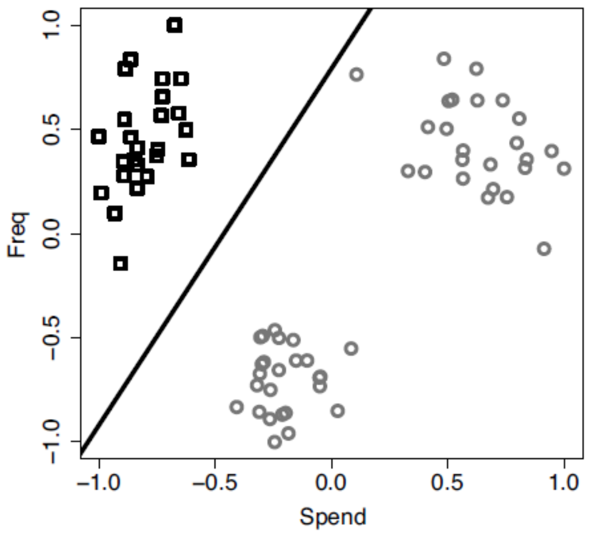
5. Multiclass Logistic Regression

- If we have r target levels (e.g., $r = 3$ classes: *single* (squares), *business* (triangles), and *family* (crosses)), we create r **one-versus-all** logit models.
- A **one-versus-all** logit model distinguishes between one level (or class) of the target feature and all the others.

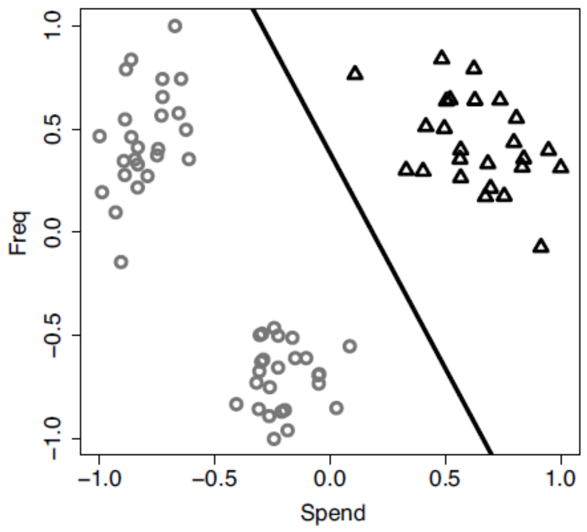
5. Multiclass Logistic Regression

- Fig. 7.20 below shows three one-versus-all prediction models for the customer type dataset *D* with three target levels (or classes): *single* (squares), *business* (triangles), and *family* (crosses). These models are based on the dataset in Table 7.11.

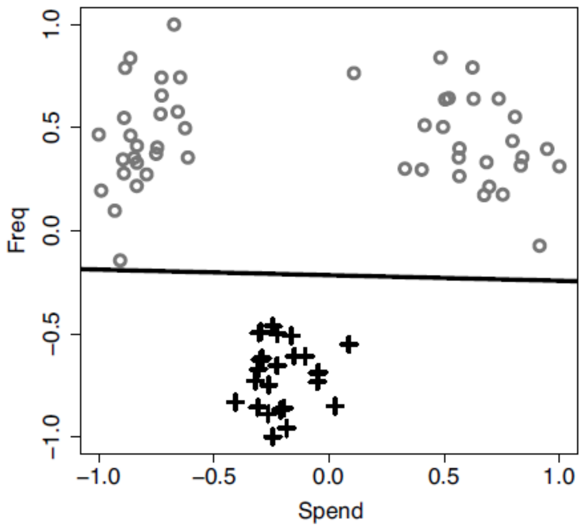
5. Multiclass Logistic Regression



(a)



(b)



(c)

- Example: $r = 3$ classes: *single* marked by squares, *business* marked by triangles, and *family* marked by crosses

5. Multiclass Logistic Regression

- **Table 7.11** A dataset *D* of customers of a large national retail chain.

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
		⋮	
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
		⋮	

(SPEND is each customer's average weekly spending, FREQ is average number of visits per week)

5. Multiclass Logistic Regression

• For r target feature levels, we build r separate **one-versus-all** logit models

$$h_{\mathbf{w}_1}(\mathbf{x}) = g(\mathbf{w}_1 \cdot \mathbf{x}),\ h_{\mathbf{w}_2}(\mathbf{x}) = g(\mathbf{w}_2 \cdot \mathbf{x}),\ \ldots,\ h_{\mathbf{w}_r}(\mathbf{x}) = g(\mathbf{w}_r \cdot \mathbf{x})$$

where

- $h_{\mathbf{w}_1}, h_{\mathbf{w}_2}, \ldots, h_{\mathbf{w}_r}$ are r different **one-versus-all** logit models
- $g(\cdot)$ is a logistic function
- $\mathbf{w}_1$ to $\mathbf{w}_r$ are r different weight vectors

5. Multiclass Logistic Regression

- $\mathbf{w}_k = (w_{k0}, w_{k1}, \ldots, w_{kd})$ (e.g., $\mathbf{w}_1 = (w_{10}, w_{11}, \ldots, w_{1d})$, $\mathbf{w}_2 = (w_{20}, w_{21}, \ldots, w_{2d})$, $\mathbf{w}_3 = (w_{30}, w_{31}, \ldots, w_{3d})$)
- $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ (or $\mathbf{x} = (1, x_1, x_2, \ldots, x_d)$)

/* recall:
- logit model: $h_{\mathbf{w}}(\mathbf{x}) = g(\mathbf{w}\cdot\mathbf{x}) = 1/(1 + e^{-\mathbf{w}\cdot\mathbf{x}}) \in (0, 1)$
- MLR model: $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$

*/

Example

- For a query instance **q** = (SPEND, FREQ) = (25.67, 6.12), which is normalized to [–1, 1], so SPEND = –0.7279 and FREQ = 0.4789, the predictions of three trained **one-versus-all** logit models are

$$h_{\mathbf{w}_{single}}(\mathbf{q}) = 0.9999,\ h_{\mathbf{w}_{business}}(\mathbf{q}) = 0.00007,\ h_{\mathbf{w}_{family}}(\mathbf{q}) = 0.01278$$

$$(\sum_{\ell=1}^{r} h_{\mathbf{w}_\ell}(\mathbf{q}) = 0.9999 + 0.00007 + 0.01278 = 1.01284 > 1)$$

5. Multiclass Logistic Regression

• To combine the outputs of r different models, we normalize their results as follows

$$h'_{\mathbf{w}_k}(\mathbf{x}) = h_{\mathbf{w}_k}(\mathbf{x}) / \sum_{\ell=1}^{r} h_{\mathbf{w}_\ell}(\mathbf{x})$$

where $h'_{\mathbf{w}_k}(\mathbf{x})$ is a **normalized prediction** for the **one-versus-all** logit model for the target level k, where $k = 1, 2, ..., r$.

• Normalization ensures that the output of all models sums to 1.

5. Multiclass Logistic Regression

- The r **one-versus-all** logit models used are trained in parallel, and the normalized model outputs $h'_{\mathbf{w}_k}(\mathbf{x})$ (i.e., $h'_{\mathbf{w}_k}(\mathbf{x}_i)$) are used when calculating the SSE for each model $h_{\mathbf{w}_k}$ during the training process.
- This means that the SSE error function L_2 of $h_{\mathbf{w}_k}$ is changed slightly to

$$L_2(h_{\mathbf{w}_k}, D) = \frac{1}{2}\sum_{i=1}^{m}(h'_{\mathbf{w}_k}(\mathbf{x}_i) - y_i)^2$$

5. Multiclass Logistic Regression

- The **normalized predictions** $h'_{\mathbf{w}_k}(\mathbf{q})$ are also used when making predictions for query instances $\mathbf{q}$.
- The predicted level for a query instance $\mathbf{q}$ is the level associated with the **one-versus-all** logit model that outputs $h'_{\mathbf{w}_k}(\mathbf{q})$ the highest normalized result. That is, we have

$$h(\mathbf{q}) = \underset{k \in \{1,2,\ldots r\}}{\arg\max}\, h'_{\mathbf{w}_k}(\mathbf{q})$$

(find value of k for which $h'_{\mathbf{w}_k}(\mathbf{q})$ is largest)

5. Multiclass Logistic Regression

- Table 7.11 shows a sample from a dataset *D* of mobile customers that includes details of customers' shopping habits with a large national retail chain.
- Each customer's average weekly spending with the chain, SPEND, and average number of visits per week to the chain, FREQ, are included along with the TYPE of customer: *single*, *business*, or *family*.

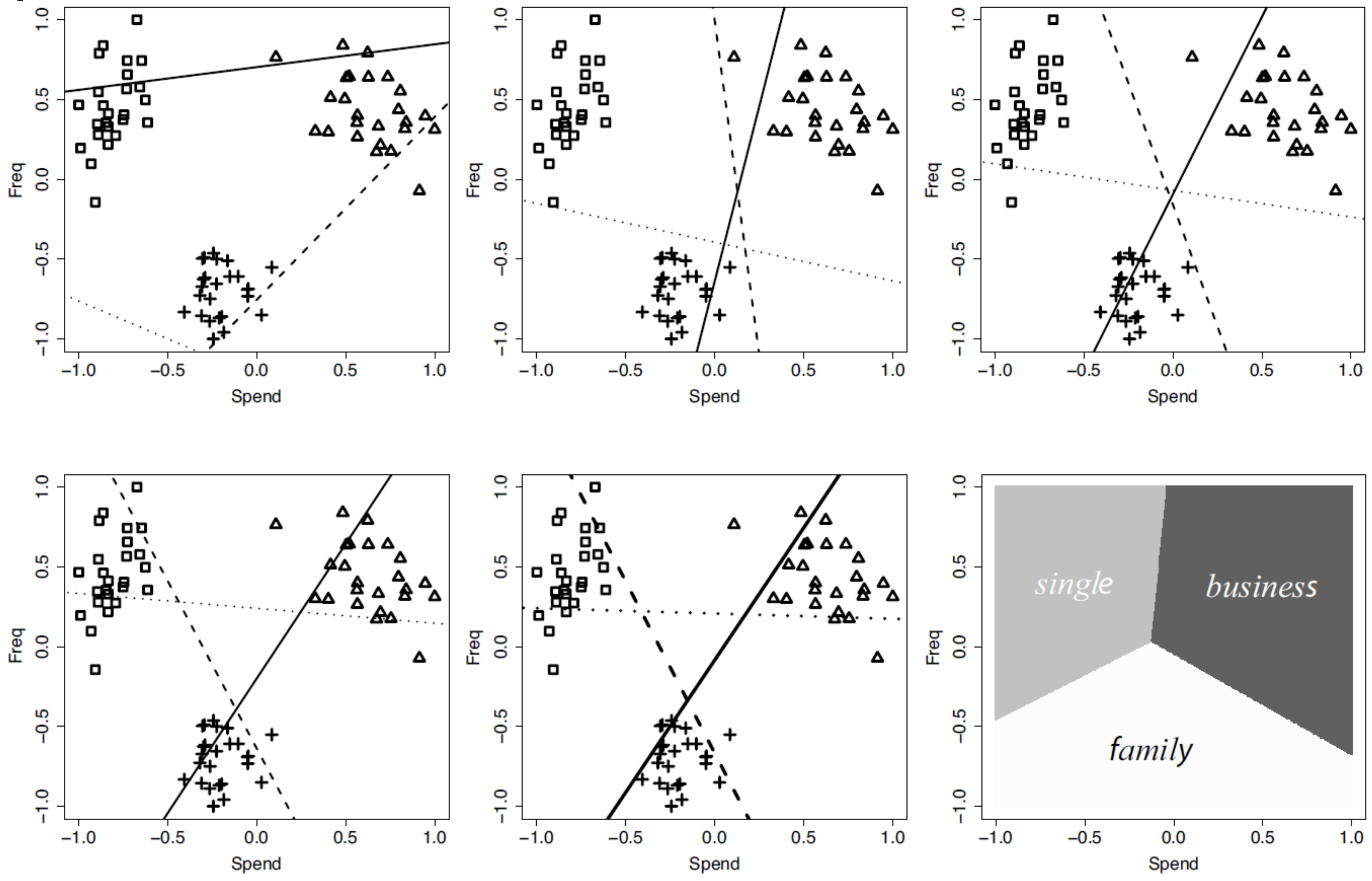
5. Multiclass Logistic Regression

• An extended version of this dataset was used to build a model that can determine the type of a customer based on a few weeks of shopping behavior data.

• Fig. 7.21 below shows the training sequence for a multiclass logit model trained using the data *D* (after data had been **range normalized** to [–1, 1]).

• There are three target levels, so three **one-versus-all** logit models are built. The evolution of the DB for each model is shown in Fig. 7.21 below.

5. Multiclass Logistic Regression



Example

- We illustrate how a prediction is made using a multiclass logit model.
- The parameters of the three trained **one-versus-all** logit models for the three final decision boundaries are

$$h_{\mathbf{w}_{single}}(\mathbf{q}) = g(0.7993 + (-15.9030) \times \text{SPEND} + 9.5974 \times \text{FREQ})$$

$$h_{\mathbf{w}_{business}}(\mathbf{q}) = g(4.6419 + 14.9401 \times \text{SPEND} + (-6.9457) \times \text{FREQ})$$

$$h_{\mathbf{w}_{family}}(\mathbf{q}) = g(3.6526 + (-0.5809) \times \text{SPEND} + (-17.5886) \times \text{FREQ})$$

Example

- For a query instance **q** = (SPEND, FREQ) = (25.67, 6.12), which are normalized to [–1, 1], so SPEND = –0.7279 and FREQ = 0.4789, the predictions of the individual models are

Example

$$h_{\mathbf{w}_{single}}(\mathbf{q}) = g(0.7993 + (-15.9030) \times (-0.7279) + 9.5974 \times 0.4789)$$
$$= 0.9999$$
$$h_{\mathbf{w}_{business}}(\mathbf{q}) = g(4.6419 + 14.9401 \times (-0.7279) + (-6.9457) \times 0.4789)$$
$$= 0.0007$$
$$h_{\mathbf{w}_{family}}(\mathbf{q}) = g(3.6526 + (-0.5809) \times (-0.7279) + (-17.5886) \times 0.4789)$$
$$= 0.01278$$

$$(\sum_{\ell=1}^{r} h_{\mathbf{w}_\ell}(\mathbf{q}) = 0.9999 + 0.00007 + 0.01278 = 1.01284 > 1)$$

Example

- These predictions are normalized as follows

$$h'_{\mathbf{w}_{single}}(\mathbf{q}) = \frac{0.9999}{0.9999 + 0.00007 + 0.01278} = \frac{0.9999}{1.01284} \approx 0.98732$$

$$h'_{\mathbf{w}_{business}}(\mathbf{q}) = \frac{0.00007}{0.9999 + 0.00007 + 0.01278} = \frac{0.00007}{1.01284} \approx 0.00007$$

$$h'_{\mathbf{w}_{family}}(\mathbf{q}) = \frac{0.01278}{0.9999 + 0.00007 + 0.01278} = \frac{0.01278}{1.01284} \approx 0.01261$$

$$\left(\sum_{k=1}^{r} h'_{\mathbf{w}_k}(\mathbf{q}) = 0.98732 + 0.00007 + 0.01261 = 1\right)$$

Example

- This means that the overall prediction for the query instance **q** is *single* as

$$h(\mathbf{q}) = \arg\max_{k \in \{1,2,\ldots r\}} h'_{\mathbf{w}_k}(\mathbf{q}) = h'_{\mathbf{w}_{single}}(\mathbf{q})$$

has the highest normalized score of ≈ 0.98 (i.e., k = 1 (*single*)).

Summary

-

Exercises

-

References

1. John D. Kelleher *et al*. 2015. *Fundamentals of Machine Learning for Predictive Data Analytics*. The MIT Press. ISBN: 0262029448.
2. Gopinath Rebala *et al*. 2019. *An Introduction to Machine Learning*. Springer. ISBN: 3030157288.
3. Shai Shalev-Shwartz, Shai Ben-David. 2014. *Understanding Machine Learning*. Cambridge University Press. ISBN: 1107057132.

References

4. Pang-Ning Tan, Michael Steinbach, Vipin Kumar. 2013. *Introduction to Data Mining*. 1st Ed (Intl). Pearson. ISBN: 1292026154.

5. Jeremy Watt *et al*. 2016. *Machine Learning Refined*. 1st Ed. Cambridge University Press. ISBN: 9781107123526. Chapter 7

6. Peter A. Flach. 2012. *Machine Learning*. Cambridge University Press. ISBN: 1107096391. Chapter 7

References

7. Simon Rogers, Mark Girolami. 2016. *A First Course in Machine Learning*. 2nd Ed. CRC Press. ISBN-10: 1498738486. Chapter 5

Extra Slides – Solving Normal Equations

-

Extra Slides

-