DATA, MSML, BIOI 602 Principles of Data Science (Fall 2024)

Linear Regression

Heng Huang
Department of Computer Science

Supervised Learning

Data: $D = \{D_1, D_2, .., D_n\}$ **a set of *n* examples**

$D_i = < \mathbf{x}_i, y_i >$

$\mathbf{x}_i = (x_{i,1}, x_{i,2}, \cdots x_{i,d})$ is an input vector of size d

y_i is the desired output (given by a teacher)

Objective: learn the mapping $f : X \rightarrow Y$

s.t. $y_i \approx f(\mathbf{x}_i)$ for all $i = 1,.., n$

- **Regression:** Y is **continuous**

  Example: earnings, product orders $\longrightarrow$ company stock price

- **Classification:** Y is **discrete**

  Example: handwritten digit in binary form $\longrightarrow$ digit label

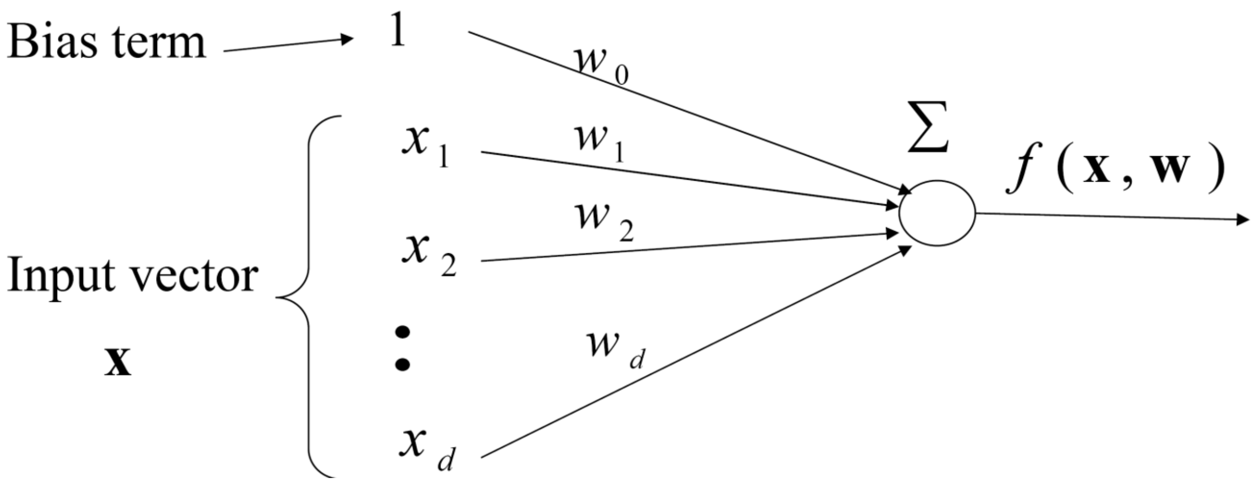
Ref: Milos Hauskrecht

Linear Regression

- **Function** $f : X \rightarrow Y$ is a linear combination of input components

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \dots w_d x_d = w_0 + \sum_{j=1}^{d} w_j x_j$$

$w_0, w_1, \dots w_k$ **- parameters (weights)**



Ref: Milos Hauskrecht

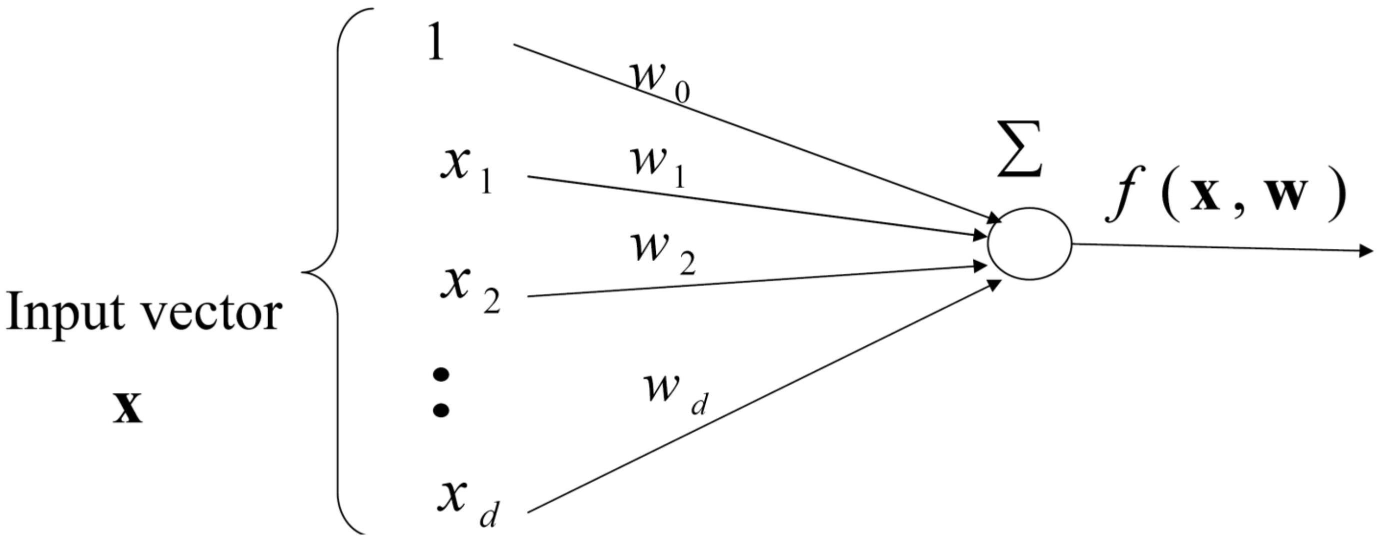
Linear Regression

- **Shorter (vector) definition of the model**
  - Include bias constant in the input vector

$$\mathbf{x} = (1, x_1, x_2, \cdots x_d)$$

$$f(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots w_d x_d = \mathbf{w}^T \mathbf{x}$$

$w_0, w_1, \ldots w_k$ **- parameters (weights)**

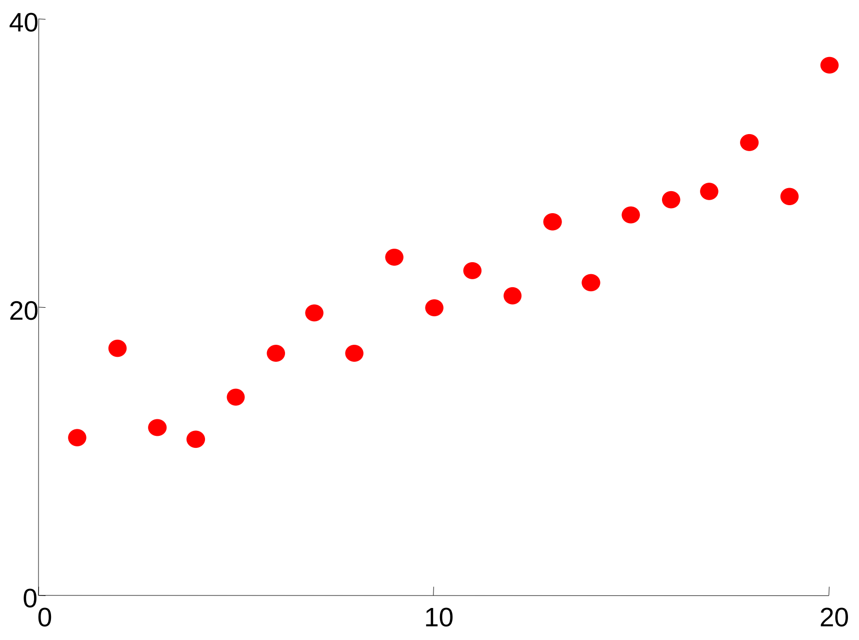


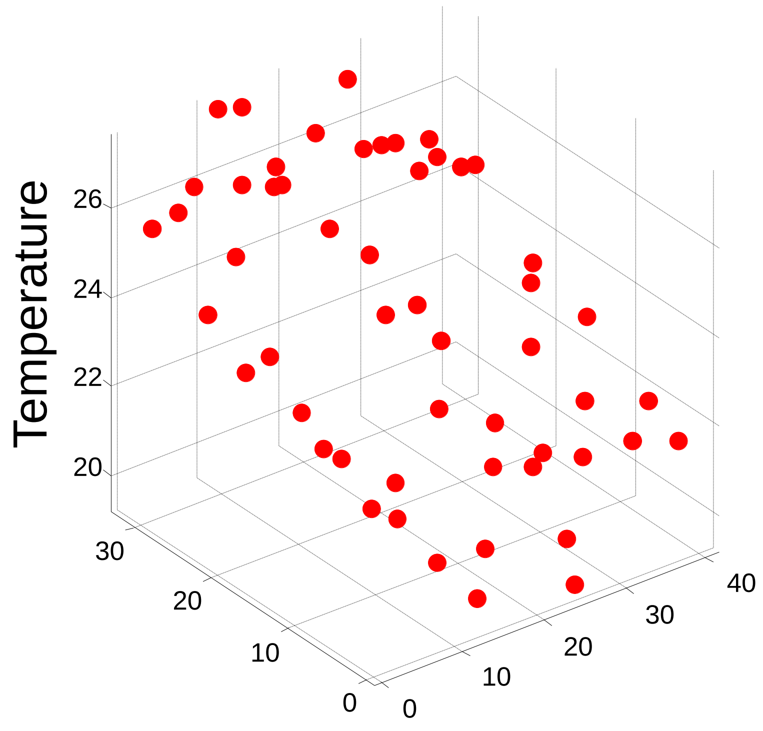
Ref: Milos Hauskrecht

Examples

- Voltage -> Temperature
- Stock prediction -> Money
- Processes, memory -> Power consumption
- Protein structure -> Energy
- Robot arm controls -> Torque at effector
- Location, industry, past losses -> Premium

Linear Regression

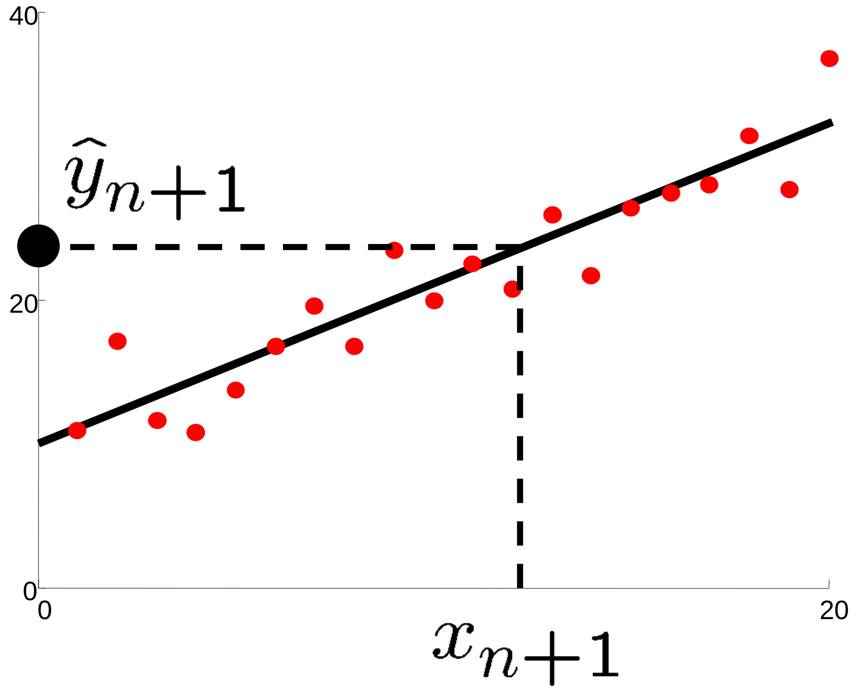


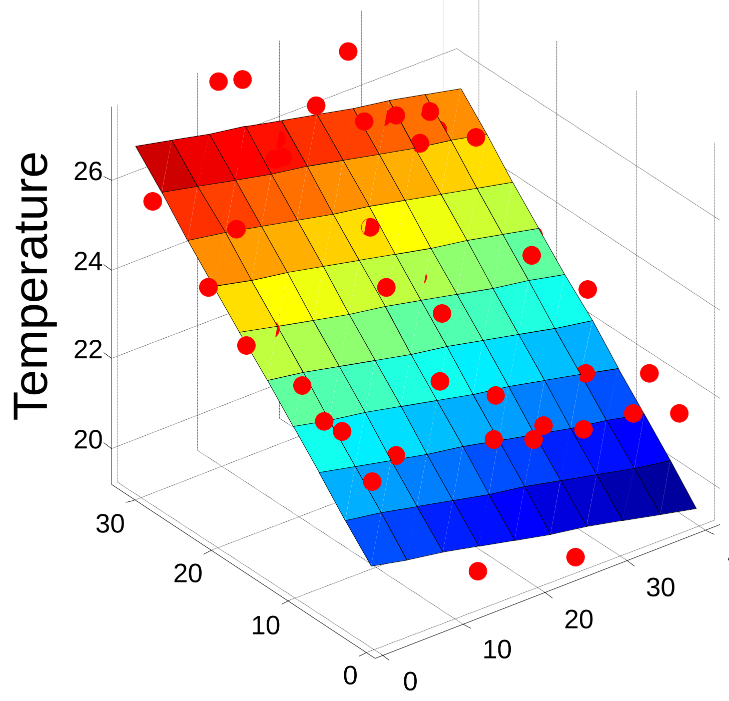


Given examples $(x_i, y_i)_{i=1 \ldots n}$

Predict y_{n+1} given a new point x_{n+1}

Linear Regression





Prediction

$$\hat{y}_i = w_0 + w_1 x_i$$

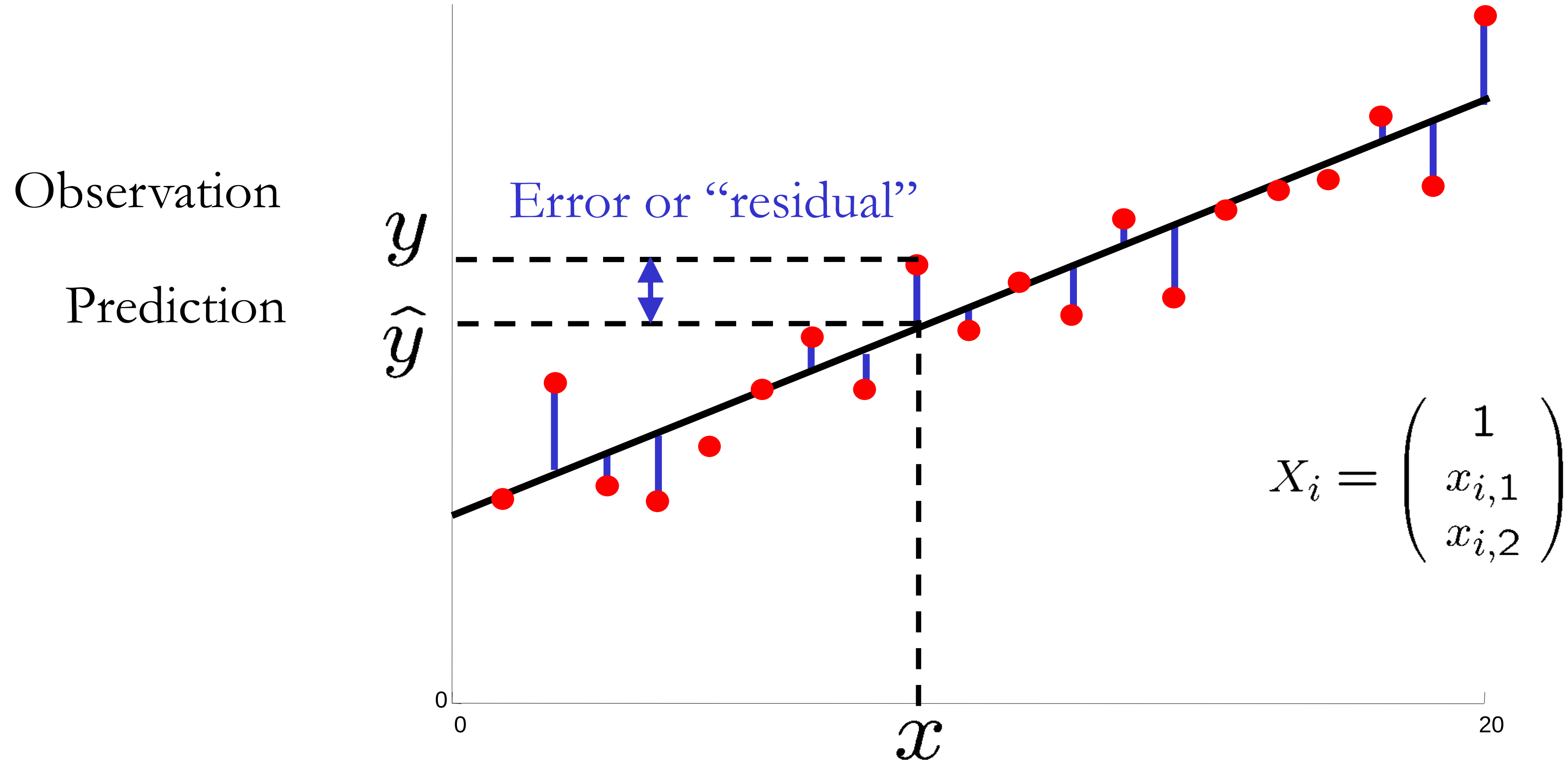
Prediction

$$\hat{y}_i = w_0 + w_1 x_{i,1} + w_2 x_{i,2}$$

$$= \begin{pmatrix} 1 & x_{i,1} & x_{i,2} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

$$= X_i^\top w$$

Ordinary Least Squares (OLS)



Sum squared error $\sum_i (X_i^\top w - y_i)^2$

Linear Regression. Optimization.

- We want the **weights minimizing the error**

$$J_n = \frac{1}{n} \sum_{i=1,..n} (y_i - f(\mathbf{x}_i))^2 = \frac{1}{n} \sum_{i=1,..n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- For the optimal set of parameters, derivatives of the error with respect to each parameter must be 0

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,j} = 0$$

- **Vector of derivatives:**

$$\text{grad}_{\mathbf{w}}(J_n(\mathbf{w})) = \nabla_{\mathbf{w}}(J_n(\mathbf{w})) = -\frac{2}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i = \overline{\mathbf{0}}$$

Ref: Milos Hauskrecht

Linear Regression. Optimization.

- $\text{grad}_{\mathbf{w}}(J_n(\mathbf{w})) = \bar{\mathbf{0}}$ defines a set of equations in **w**

$$\frac{\partial}{\partial w_0} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) = 0$$

$$\frac{\partial}{\partial w_1} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,1} = 0$$

$$\ldots$$

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,j} = 0$$

$$\ldots$$

$$\frac{\partial}{\partial w_d} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d}) x_{i,d} = 0$$

Ref: Milos Hauskrecht

Solving Linear Regression

$$\frac{\partial}{\partial w_j} J_n(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - w_0 x_{i,0} - w_1 x_{i,1} - \ldots - w_d x_{i,d})x_{i,j} = 0$$

By rearranging the terms we get a **system of linear equations** with *d+1* unknowns

$$\boxed{\mathbf{A}\mathbf{w} = \mathbf{b}}$$

$$w_0 \sum_{i=1}^{n} x_{i,0} 1 + w_1 \sum_{i=1}^{n} x_{i,1} 1 + \ldots + w_j \sum_{i=1}^{n} x_{i,j} 1 + \ldots + w_d \sum_{i=1}^{n} x_{i,d} 1 = \sum_{i=1}^{n} y_i 1$$

$$w_0 \sum_{i=1}^{n} x_{i,0} x_{i,1} + w_1 \sum_{i=1}^{n} x_{i,1} x_{i,1} + \ldots + w_j \sum_{i=1}^{n} x_{i,j} x_{i,1} + \ldots + w_d \sum_{i=1}^{n} x_{i,d} x_{i,1} = \sum_{i=1}^{n} y_i x_{i,1}$$

$$\bullet\bullet\bullet$$

$$w_0 \sum_{i=1}^{n} x_{i,0} x_{i,j} + w_1 \sum_{i=1}^{n} x_{i,1} x_{i,j} + \ldots + w_j \sum_{i=1}^{n} x_{i,j} x_{i,j} + \ldots + w_d \sum_{i=1}^{n} x_{i,d} x_{i,j} = \sum_{i=1}^{n} y_i x_{i,j}$$

$$\bullet\bullet\bullet$$

Ref: Milos Hauskrecht

Solving Linear Regression

- The optimal set of weights satisfies:

$$\nabla_{\mathbf{w}}(J_n(\mathbf{w})) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^T\mathbf{x}_i)\mathbf{x}_i = \bar{\mathbf{0}}$$

Leads to a **system of linear equations (SLE)** with *d+1* unknowns of the form

$$\mathbf{A}\mathbf{w} = \mathbf{b}$$

$$w_0\sum_{i=1}^{n}x_{i,0}x_{i,j} + w_1\sum_{i=1}^{n}x_{i,1}x_{i,j} + \ldots + w_j\sum_{i=1}^{n}x_{i,j}x_{i,j} + \ldots + w_d\sum_{i=1}^{n}x_{i,d}x_{i,j} = \sum_{i=1}^{n}y_i x_{i,j}$$

Solution to SLE:

$$\boxed{\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}}$$

Ref: Milos Hauskrecht

Gradient Descent Solution

Goal: the weight optimization in the linear regression model

$$J_n = Error(\mathbf{w}) = \frac{1}{n} \sum_{i=1,..n} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

An alternative to SLE solution:

- **Gradient descent**

  Idea:

  – Adjust weights in the direction that improves the Error

  – The gradient tells us what is the right direction

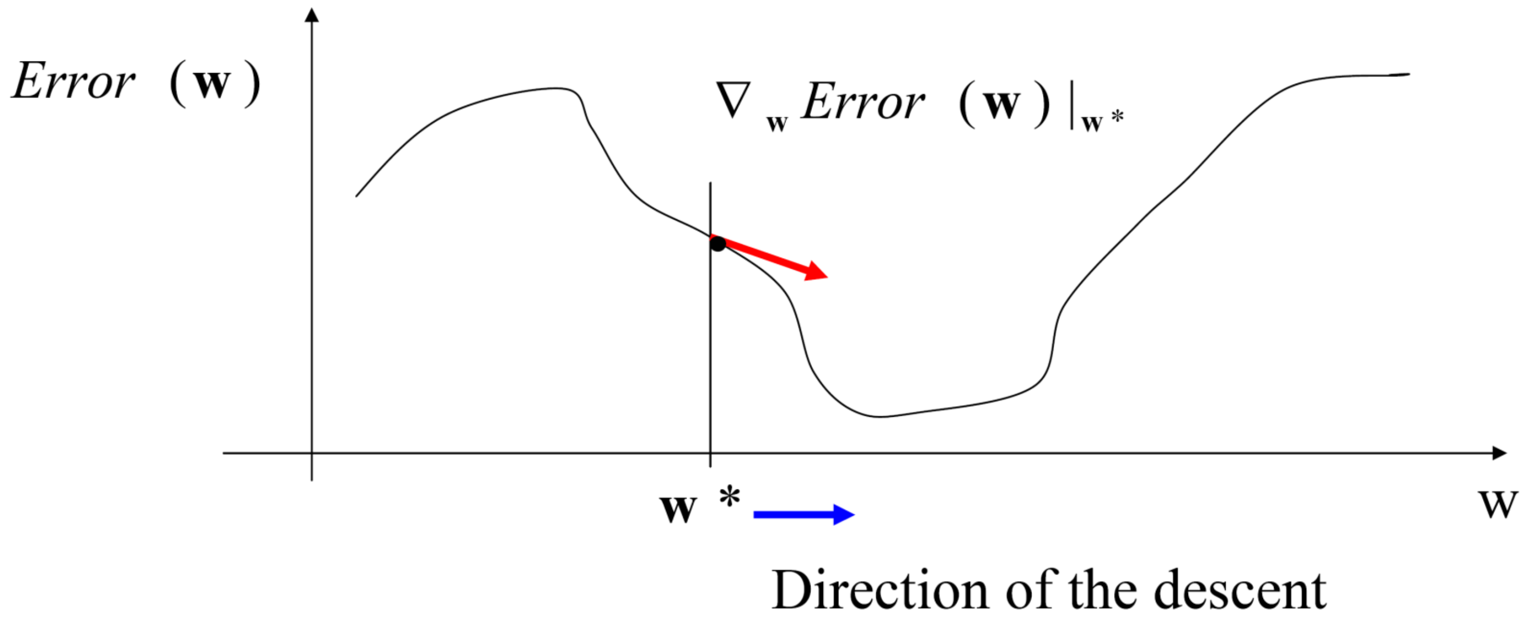
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} Error_i(\mathbf{w})$$

$\alpha > 0$ - a **learning rate** (scales the gradient changes)

Ref: Milos Hauskrecht

Gradient Descent Method

- Descend using the gradient information

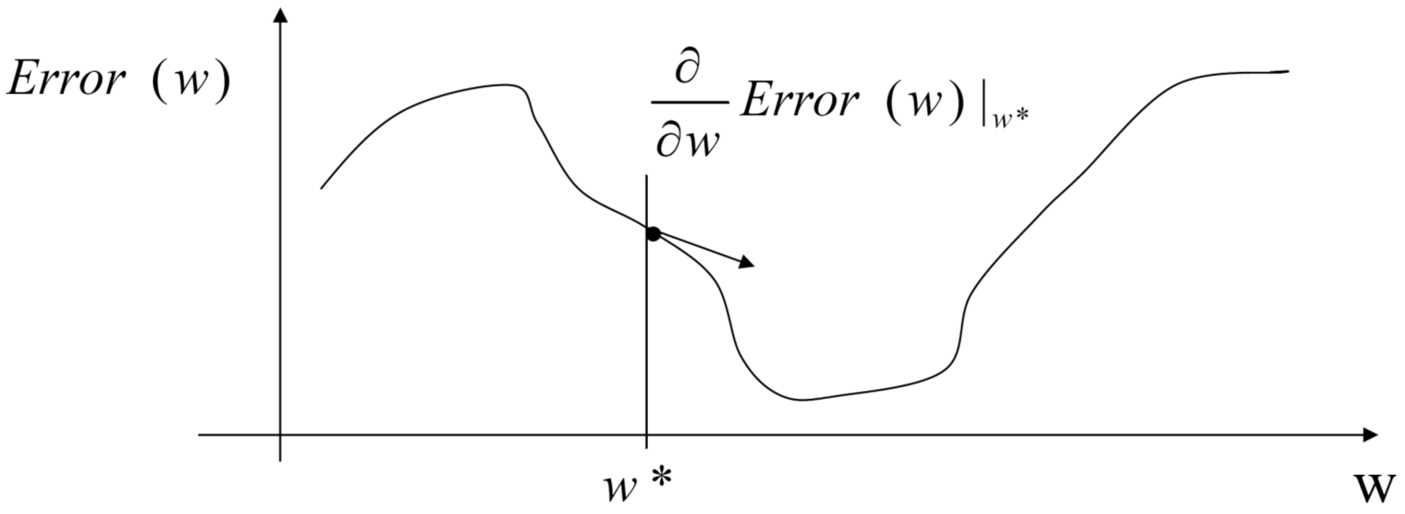


- Change the value of **w** according to the gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} Error_i(\mathbf{w})$$

Ref: Milos Hauskrecht

Gradient Descent Method



- New value of the parameter

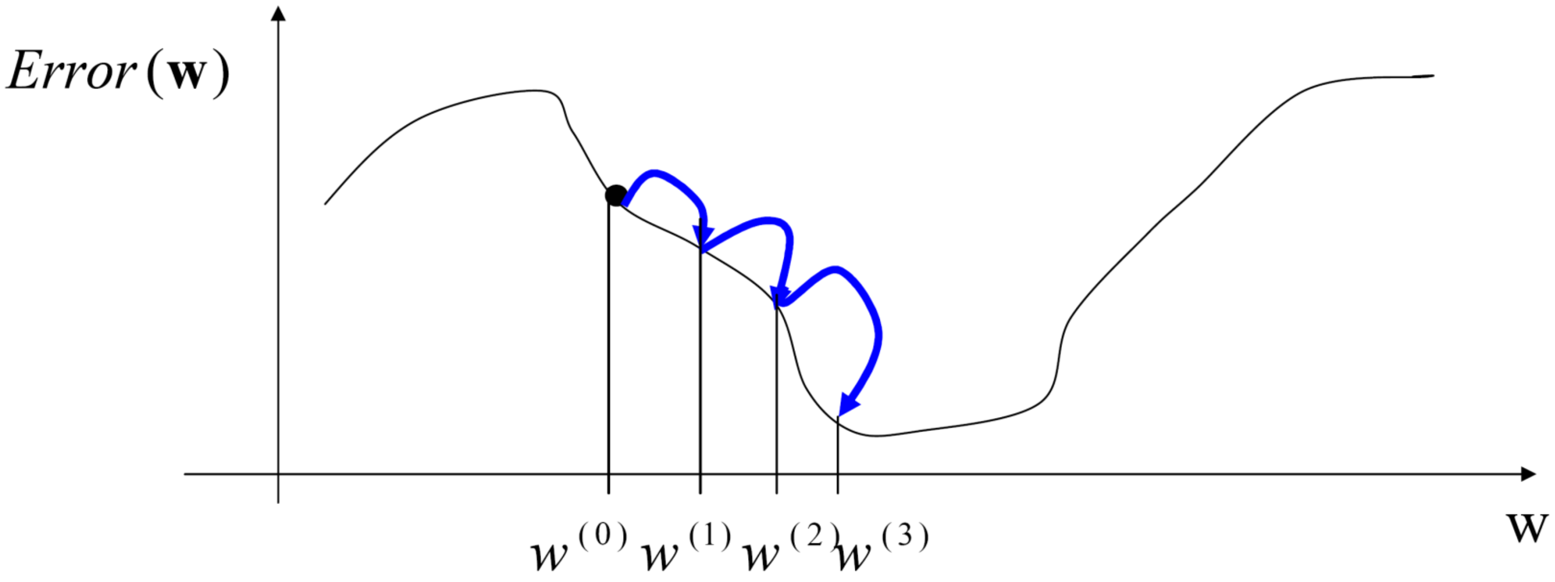
$$w_j \leftarrow w_j^* - \alpha \frac{\partial}{\partial w_j} Error(w)|_{w^*} \qquad \text{For all j}$$

$\alpha > 0$ - a learning rate (scales the gradient changes)

Ref: Milos Hauskrecht

Gradient Descent Method

- Iteratively approaches the optimum of the Error function



Ref: Milos Hauskrecht

Online Gradient Algorithm

Linear model $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

On-line error $J_{online} = Error_i(\mathbf{w}) = \frac{1}{2}(y_i - f(\mathbf{x}_i, \mathbf{w}))^2$

On-line algorithm: generates a sequence of online updates

(i)-th update step with : $D_i =< \mathbf{x}_i, y_i >$

j-th weight:

$$w_j^{(i)} \leftarrow w_j^{(i-1)} - \alpha(i) \frac{\partial Error_i(\mathbf{w})}{\partial w_j} \Big|_{\mathbf{w}^{(i-1)}}$$

$$\boxed{w_j^{(i)} \leftarrow w_j^{(i-1)} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}^{(i-1)}))x_{i,j}}$$

Fixed learning rate: $\alpha(i) = C$

- Use a small constant

Annealed learning rate: $\alpha(i) \approx \frac{1}{i}$

- Gradually rescales changes

Ref: Milos Hauskrecht

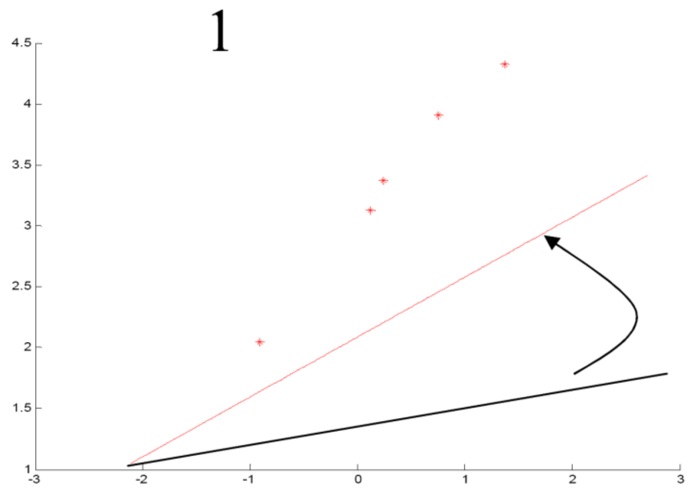
Online Regression Algorithm

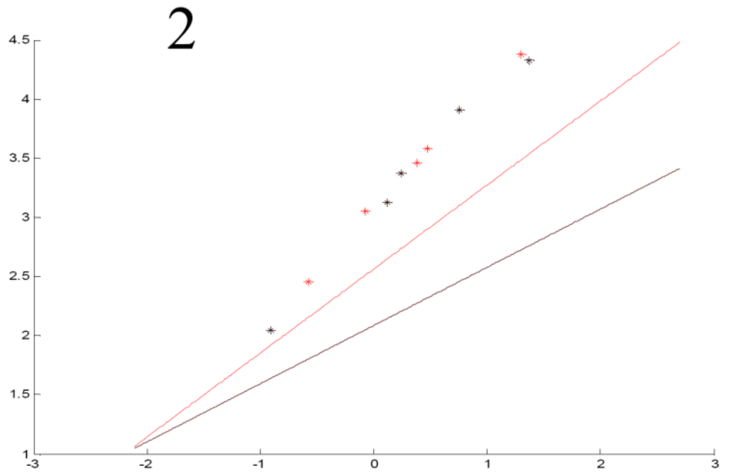
```
**Online-linear-regression** (D, number of iterations)
    Initialize weights $\mathbf{w} = (w_0, w_1, w_2 \ldots w_d)$
    for $i$=1:1: number of iterations
        do    select a data point $D_i = (\mathbf{x}_i, y_i)$ from $D$
              set learning rate $\alpha(i)$
              update weight vector
                  $\mathbf{w} \leftarrow \mathbf{w} + \alpha(i)(y_i - f(\mathbf{x}_i, \mathbf{w}))\mathbf{x}_i$
    end for
    return weights $\mathbf{w}$
```

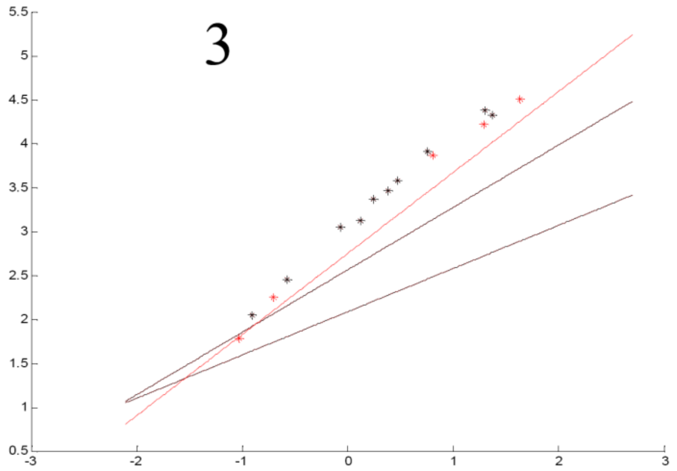
- **Advantages:** very easy to implement, continuous data streams

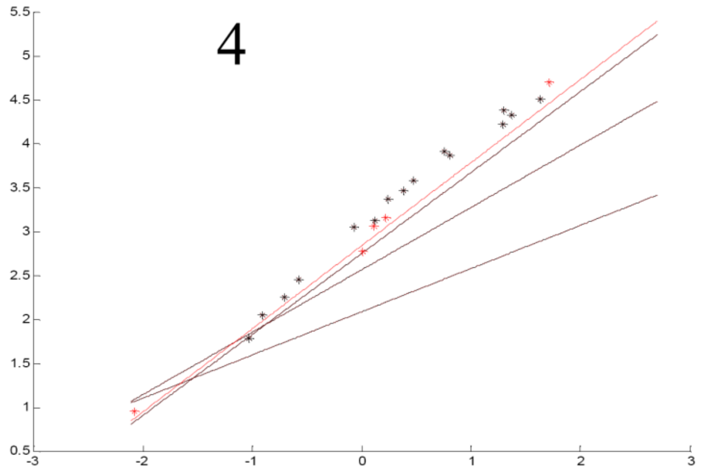
Ref: Milos Hauskrecht

On-line Learning Example









Ref: Milos Hauskrecht

Solving Linear Regression for Classification

$$\text{Input:} X \in \Re^{d\times n}, Y \in \Re^{c\times n}, W \in \Re^{d\times c}$$

$$\begin{aligned}
J &= \min_W ||Y - W^T X||_F^2 && (1)\\
&= \min_W Tr((Y - W^T X)^T (Y - W^T X)) && (2)\\
&= \min_W Tr((Y^T - X^T W)(Y - W^T X)) && (3)\\
&= \min_W Tr(Y^T Y - Y^T W^T X - X^T W Y + X^T W W^T X) && (4)\\
&\quad \text{Taking } \frac{\partial J}{\partial W} = 0
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{X}} \mathrm{Tr}(\mathbf{XA}) &= \mathbf{A}^T\\
\frac{\partial}{\partial \mathbf{X}} \mathrm{Tr}(\mathbf{AXB}) &= \mathbf{A}^T\mathbf{B}^T\\
\frac{\partial}{\partial \mathbf{X}} \mathrm{Tr}(\mathbf{AX}^T\mathbf{B}) &= \mathbf{BA}\\
\frac{\partial}{\partial \mathbf{X}} \mathrm{Tr}(\mathbf{X}^T\mathbf{A}) &= \mathbf{A}\\
\frac{\partial}{\partial \mathbf{X}} \mathrm{Tr}(\mathbf{AX}^T) &= \mathbf{A}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial J}{\partial W} &= 0 - XY^T - (YX^T)^T + (YX^T)^T + XX^T W\\
&= -XY^T - XY^T + XX^T W + XX^T W\\
&= -2XY^T + 2XX^T W
\end{aligned}$$

Solving Linear Regression for Classification

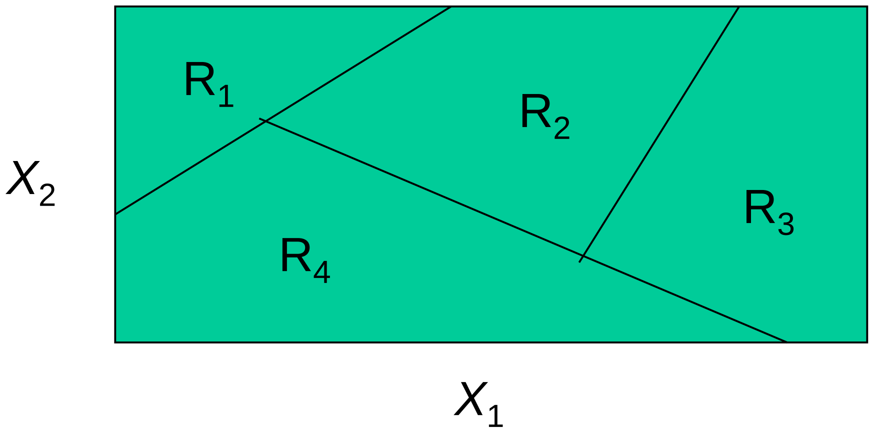
$$
\begin{aligned}
& \frac{\partial J}{\partial W} = 0 \\
\Rightarrow \quad & XY^T - XX^TW = 0 \\
\Rightarrow \quad & W = (XX^T)^{-1}XY^T
\end{aligned}
$$

- Linear ridge regression:

$$
\begin{aligned}
J \quad &= \quad \min_W ||Y - W^TX||_F^2 + \lambda||W||_F^2 \\
&\Rightarrow \quad W = (XX^T + \lambda I)^{-1}XY^T
\end{aligned}
$$

Linear Classification Discriminant Function

- There is a discriminant function $\delta_k(x)$ for each class k
- Classification rule: $R_k = \{x : k = \arg\max_j \delta_j(x)\}$
- In higher dimensional space the decision boundaries are piecewise hyperplanar



Linear Classification as A Linear Regression

2D Input space: $X = (X_1, X_2)$

Number of classes/categories K=3, So output $Y = (Y_1, Y_2, Y_3)$

Training sample, size N=5,

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ 1 & x_{31} & x_{32} \\ 1 & x_{41} & x_{42} \\ 1 & x_{51} & x_{52} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ \boxed{y_{31} \quad y_{32} \quad y_{33}} \\ y_{41} & y_{42} & y_{43} \\ y_{51} & y_{52} & y_{53} \end{bmatrix}$$

Each row has exactly one 1 indicating the category/class

Indicator Matrix

Regression output: $\hat{Y}((x_1, x_2)) = (1\ x_1\ \ x_2)(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = (x^T\beta_1\ \ x^T\beta_2\ \ x^T\beta_3)$

Bishop (3.35)

Or,

$$\hat{Y}_1((x_1\ x_2)) = (1\ x_1\ \ x_2)\beta_1$$
$$\hat{Y}_2((x_1\ x_2)) = (1\ x_1\ \ x_2)\beta_2$$
$$\hat{Y}_3((x_1\ x_2)) = (1\ x_1\ \ x_2)\beta_3$$

Classification rule:

$$\hat{G}((x_1\ \ x_2)) = \arg\max_k \hat{Y}_k((x_1\ \ x_2))$$

Diabetes Data Example

- **Diabetes data**

  The diabetes data set is taken from the UCI machine learning database repository at: http://www.ics.uci.edu/~mlearn/Machine-Learning.html .

  The original source of the data is the National Institute of Diabetes and Digestive and Kidney Diseases. There are 768 cases in the data set, of which 268 show signs of diabetes according to World Health organization criteria. Each case contains 8 quantitative variables, including diastolic blood pressure, triceps skin fold thickness, a body mass index, etc.

  – Two classes: with or without signs of diabetes.

  – Denote the 8 original variables by $\tilde{X}_1$, $\tilde{X}_2$, ..., $\tilde{X}_8$

  – Remove the mean of $\tilde{X}_j$ and normalize it to unit variance.

Principle Features

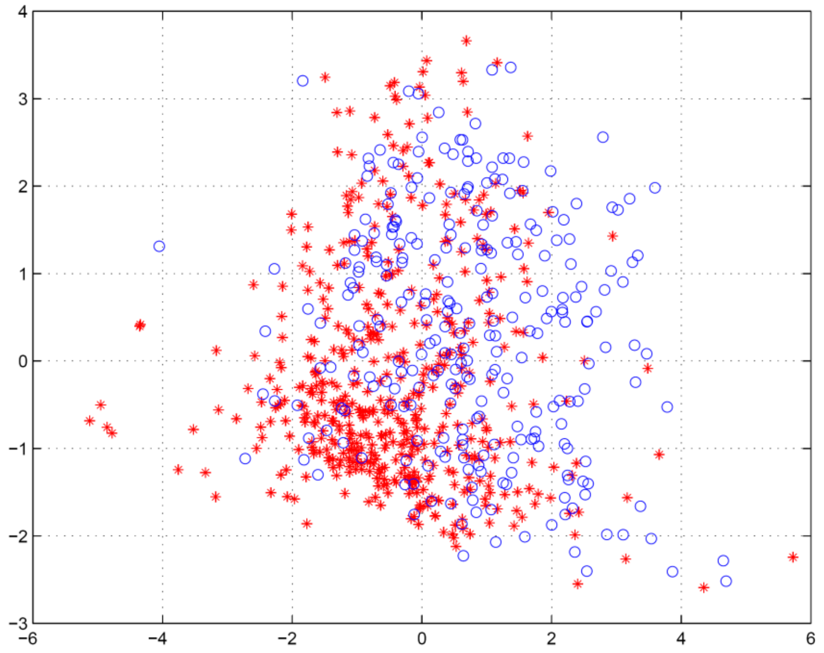
- The two principal components $X1$ and $X2$ are used in classification:

$$\begin{aligned} X_1 &= 0.1284\tilde{X}_1 + 0.3931\tilde{X}_2 + 0.3600\tilde{X}_3 + 0.4398\tilde{X}_4 \\ &\quad +0.4350\tilde{X}_5 + 0.4519\tilde{X}_6 + 0.2706\tilde{X}_7 + 0.1980\tilde{X}_8 \end{aligned}$$

$$\begin{aligned} X_2 &= 0.5938\tilde{X}_1 + 0.1740\tilde{X}_2 + 0.1839\tilde{X}_3 - 0.3320\tilde{X}_4 \\ &\quad -0.2508\tilde{X}_5 - 0.1010\tilde{X}_6 - 0.1221\tilde{X}_7 + 0.6206\tilde{X}_8 \end{aligned}$$

Linear Regression for Classification

- The scatter plot follows. Without diabetes: stars (class 1), with diabetes: circles (class 2).



$$\hat{\mathbf{B}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$
$$= \begin{pmatrix} 0.6510 & 0.3490 \\ -0.1256 & 0.1256 \\ -0.0729 & 0.0729 \end{pmatrix}$$

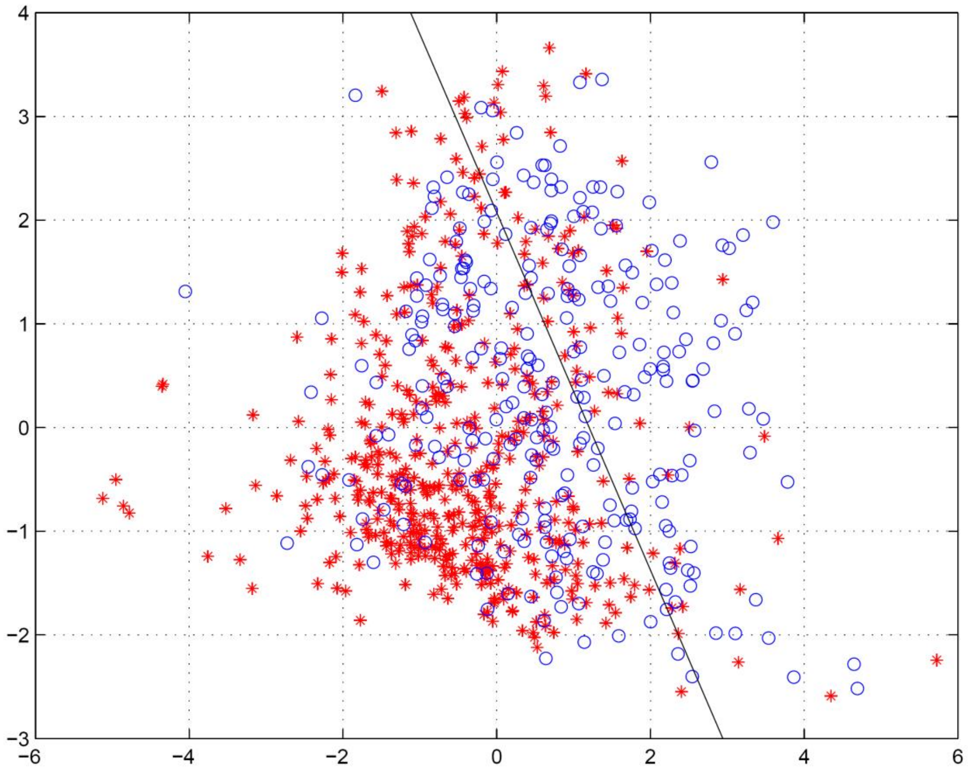
$$\hat{Y}_1 = 0.6510 - 0.1256X_1 - 0.0729X_2$$
$$\hat{Y}_2 = 0.3490 + 0.1256X_1 + 0.0729X_2$$

Classification Rule

$$\hat{G}(x) = \begin{cases} 1 & \hat{Y}_1 \geq \hat{Y}_2 \\ 2 & \hat{Y}_1 < \hat{Y}_2 \end{cases}$$

$$= \begin{cases} 1 & 0.151 - 0.1256X_1 - 0.0729X_2 \geq 0 \\ 2 & otherwise \end{cases}$$

- Classification error rate: 28.52%.



4
3
2
1
0
−1
−2
−3
−6
−4
−2
0
2
4
6