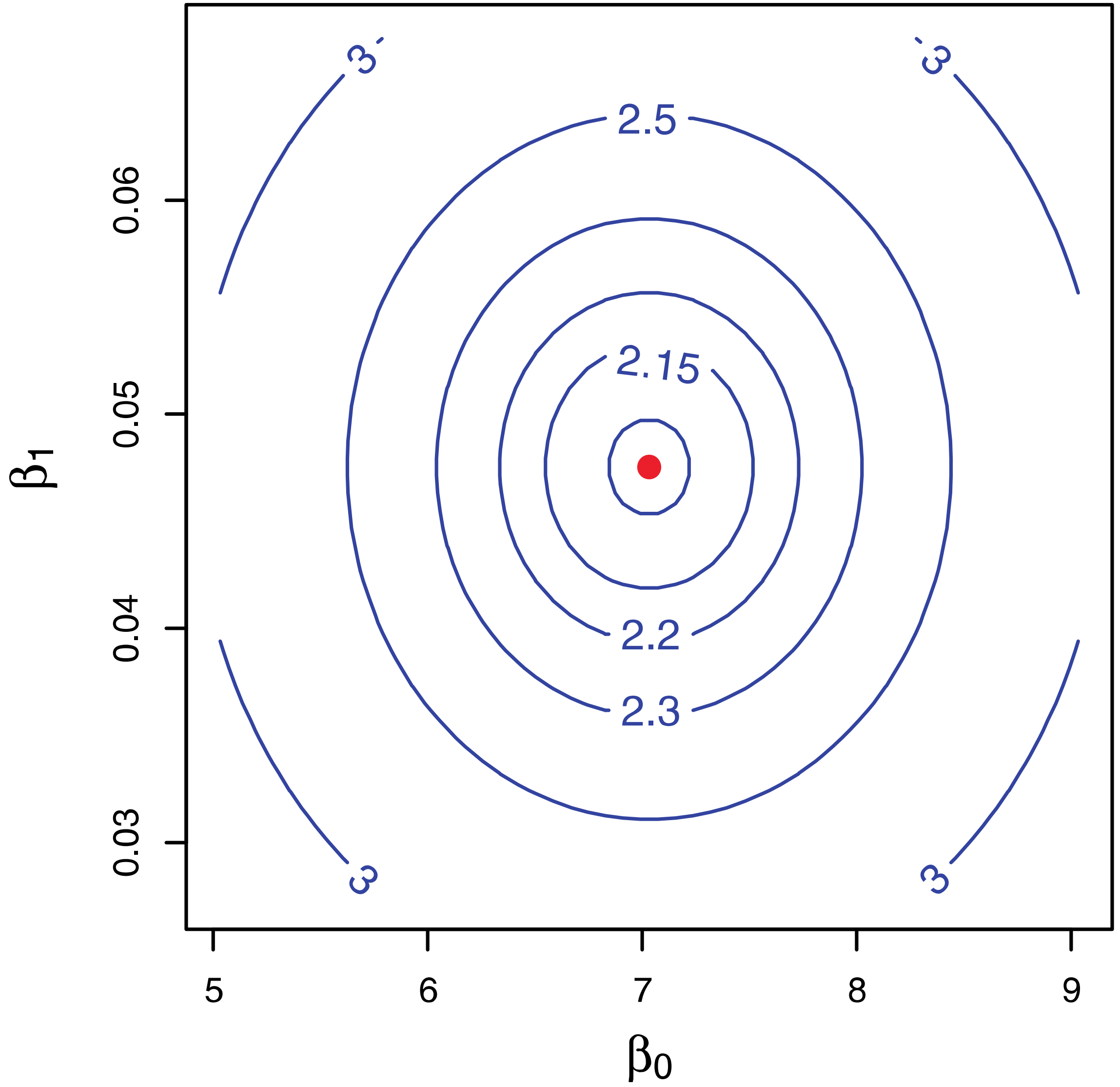
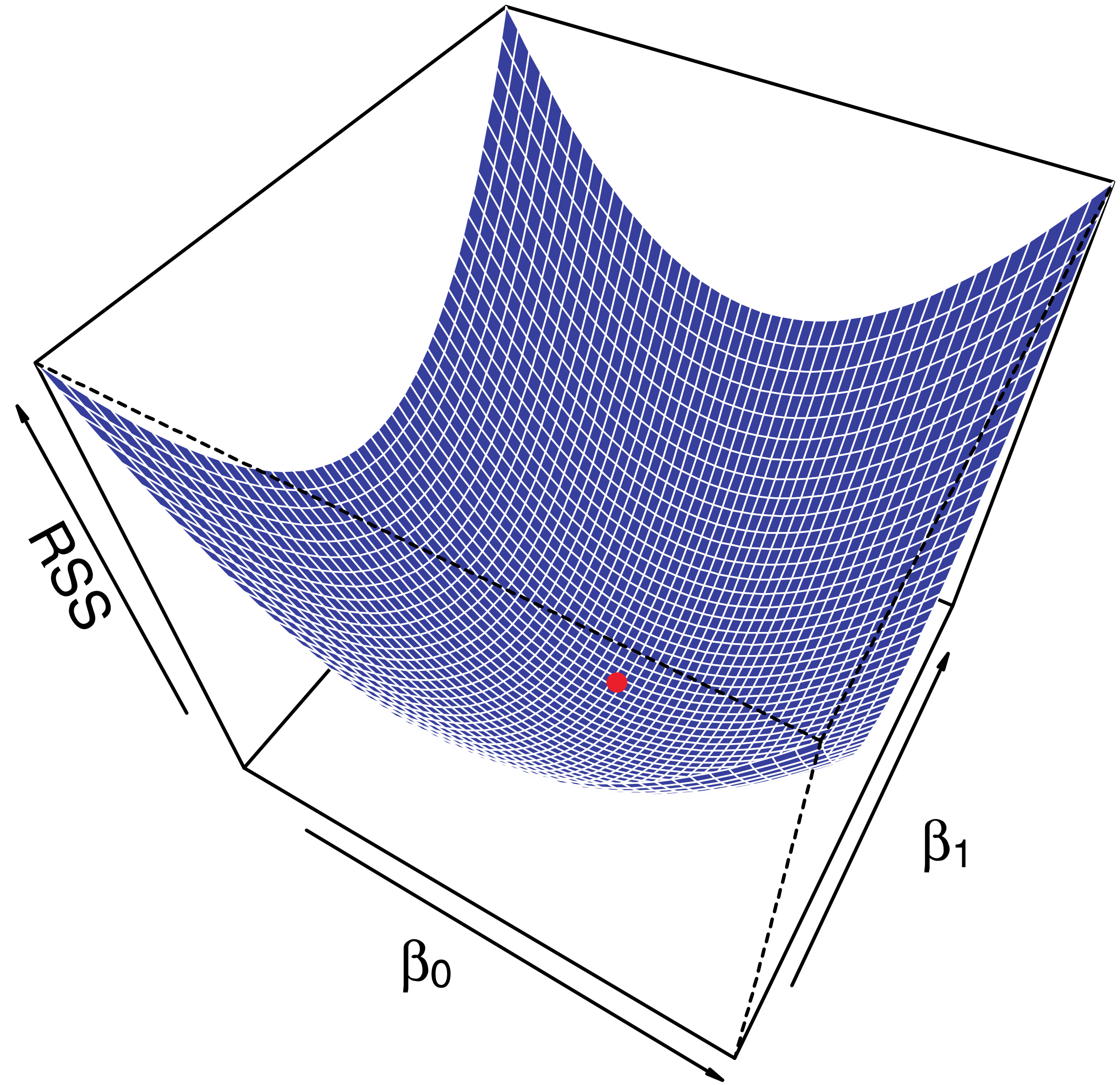
Linear models I

Lecture 06

Quiz



3
3
2.5
2.15
2.2
2.3
3
3
0.06
0.05
0.04
0.03
β_1
5
6
7
8
9
β_0



RSS
β_0
β_1

What is a linear model?

Which of the following models are linear?

A $y = w_0$

B $y = w_0 + w_1 x_1$

C $y = w_0 + w_1 x_1 + w_2 x_2$

D $y = w_0 + w_1 x_1^2 + w_2 x_2^{0.4}$

E $y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2$

F $y = w_0 + w_1 \int \sqrt[3]{x_1} dx_1 + w_2 g(x_2) + w_3 median(x_1, x_2, x_3)$

Which of the following models are linear?

A $y = w_0$

These are **ALL** linear in the **parameters, *w***

B $y = w_0 + w_1 x_1$

C $y = w_0 + w_1 x_1 + w_2 x_2$

D $y = w_0 + w_1 x_1^2 + w_2 x_2^{0.4}$

E $y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2$

F $y = w_0 + w_1 \int \sqrt[3]{x_1}\, dx_1 + w_2 g(x_2) + w_3 median(x_1, x_2, x_3)$

Linear models are linear in the **parameters**

Linear models are linear in the parameters

They often model nonlinear relationships between features and targets

$$y_j = \sum_{i=0}^{N} w_i x_{i,j} + \epsilon$$

Linear regression assumptions

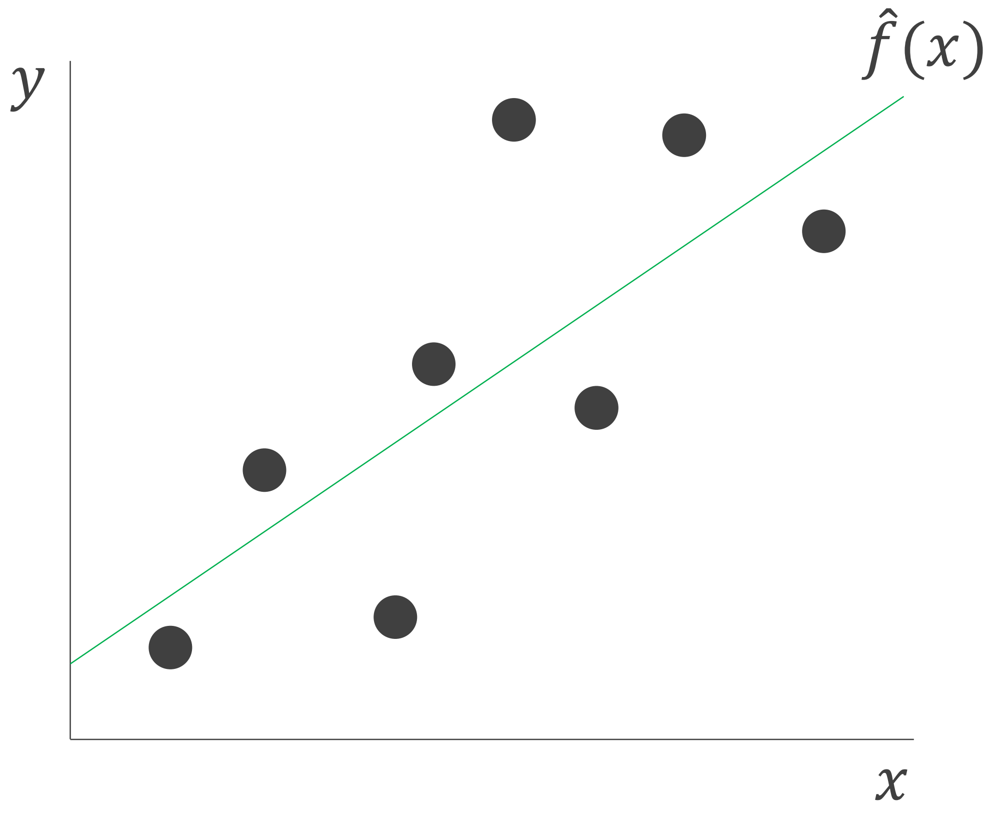
1. Linear relationship between feature and target variables
2. Error is normally distributed
3. Features are not correlated with one another (no multicollinearity)
4. Assumes observations are independent from one another (no autocorrelation)
5. Variance of the error is constant (homoscedastic)

Types of Linear Regression

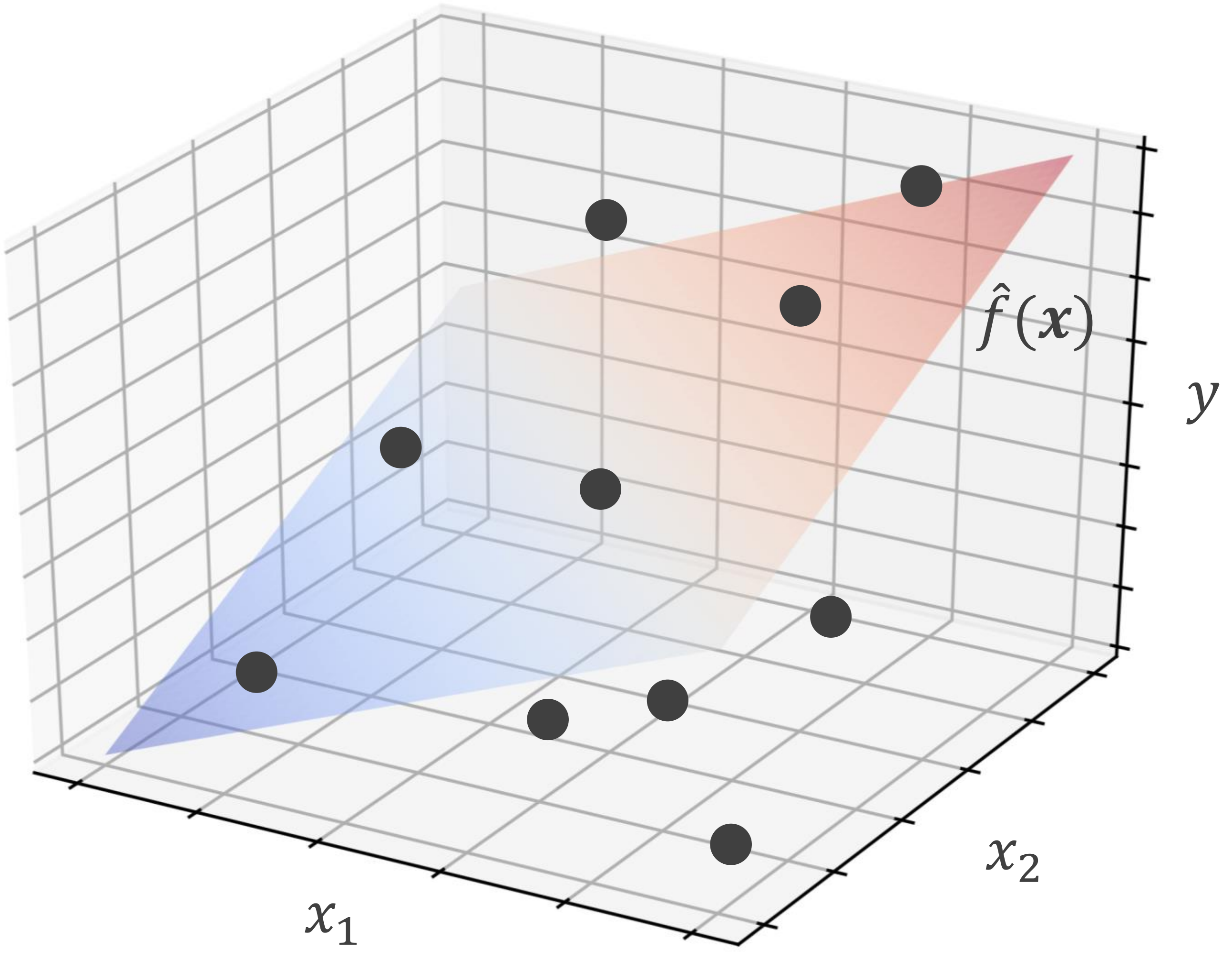
Here, $x_0 \triangleq 1$

	One feature variable	**One or more** feature variables
One target variable	**Simple Linear Regression** $y = w_0 + w_1 x_1$	**Multiple Linear Regression** $y = \sum_{i=0}^{N} w_i x_i$ or $y = \boldsymbol{w}^T \boldsymbol{x}$
One or more target variables	**Multivariate (Multiple) Linear Regression** $\boldsymbol{y} = \sum_{i=0}^{p} w_i \boldsymbol{x}_i$ or $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}$	

Linear models and error

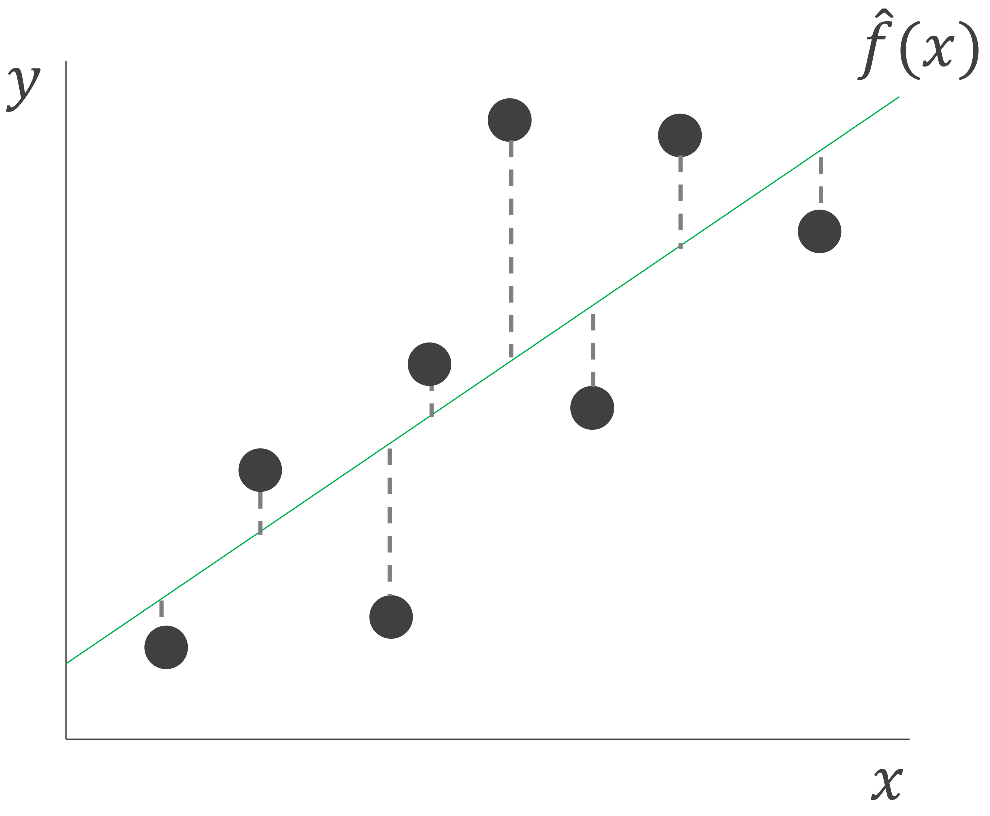


simple linear regression

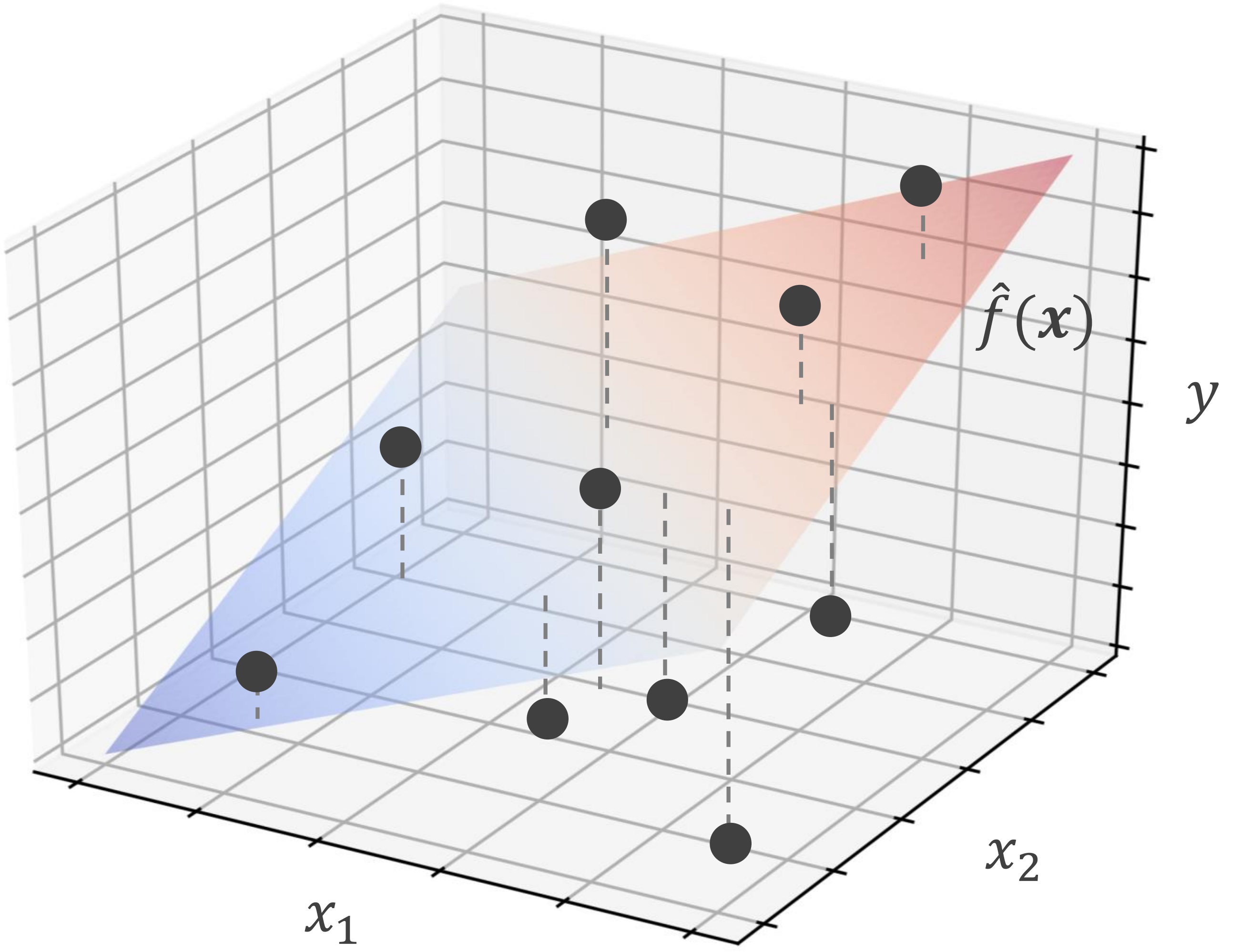


multiple linear regression

Linear models and error

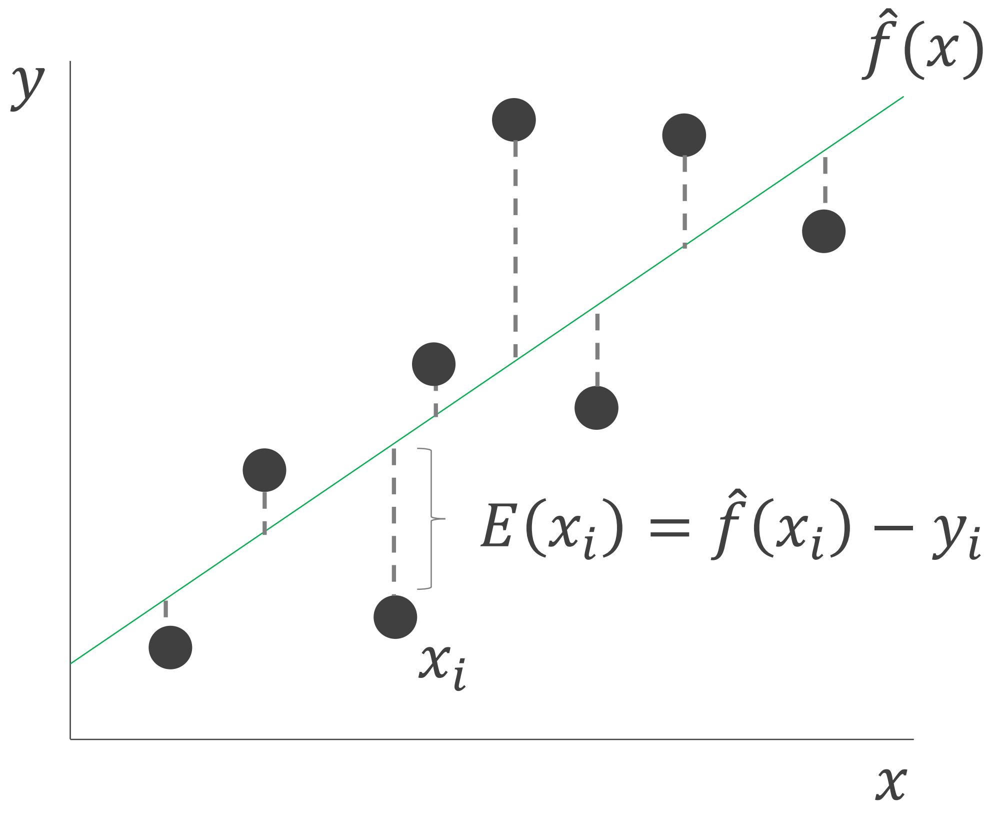


simple linear regression

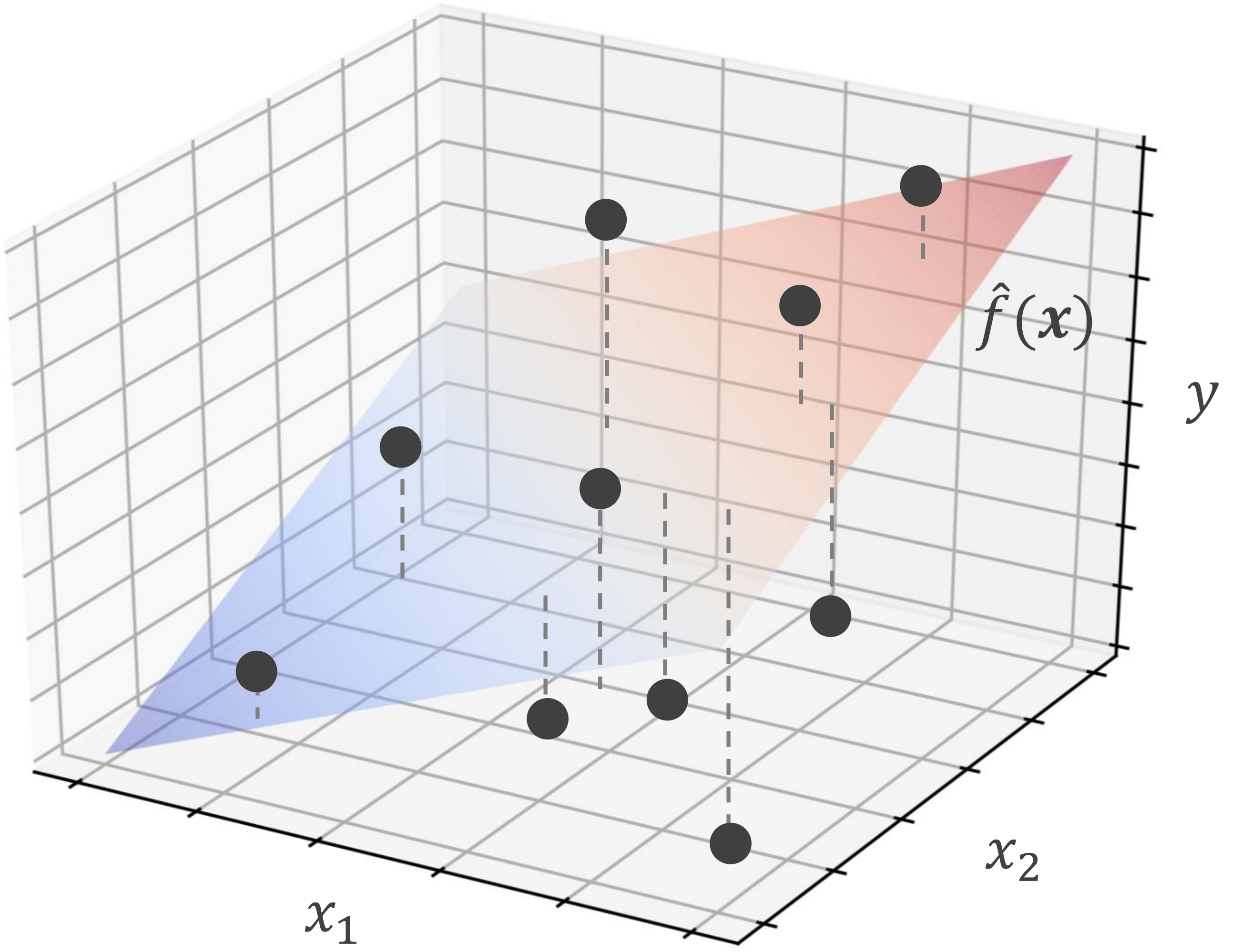


multiple linear regression

Linear models and error



simple linear regression



multiple linear regression

How do we fit a linear model to data?

We want the error between our estimates and predictions to be small

How do we measure error?

How well does $\hat{y} = \hat{f}(\boldsymbol{x}) = \sum_{i=0}^{N} w_i x_i$ approximate y ?

How do we measure error?

How well does $\hat{y} = \hat{f}(\boldsymbol{x}) = \sum_{i=0}^{N} w_i x_i$ approximate y ?

Error: difference between our estimate $\hat{y}$ and our training data y

$$\text{error} = \hat{y} - y$$

How do we measure error?

How well does $\hat{y} = \hat{f}(\boldsymbol{x}) = \sum_{i=0}^{N} w_i x_i$ approximate y ?

Error: difference between our estimate $\hat{y}$ and our training data y

$$\text{error} = \hat{y} - y$$

We use mean squared error to estimate training (in-sample) error:

Training (in-sample) error: $E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$

How do we measure error?

How well does $\hat{y} = \hat{f}(\boldsymbol{x}) = \sum_{i=0}^{N} w_i x_i$ approximate y ?

Error: difference between our estimate $\hat{y}$ and our training data y

$$\text{error} = \hat{y} - y$$

We use mean squared error to estimate training (in-sample) error:

Training (in-sample) error: $$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

We call this our **Cost Function**

Cost Function: $$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

Cost Function: $$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

Training error is a function of our **model** and the **training data**

Cost Function: $$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

Training error is a function of our **model** and the **training data**

We can't change the data, we must adjust our model to minimize cost

Cost Function: $$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

Training error is a function of our **model** and the **training data**

We can’t change the data, we must adjust our model to minimize cost

We choose model **parameters** that minimize cost

Cost Function: $$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

Training error is a function of our **model** and the **training data**

We can't change the data, we must adjust our model to minimize cost

We choose model **parameters** that minimize cost

This is an **optimization** problem

How to fit our model to the training data?

Equivalently: how do we choose $\boldsymbol{w}$ to minimize cost (error)

$$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

How to fit our model to the training data?

Equivalently: how do we choose $\boldsymbol{w}$ to minimize cost (error)

$$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

where $\hat{f}(\boldsymbol{x}_n) = \boldsymbol{w}^T\boldsymbol{x}_n$

How to fit our model to the training data?

Equivalently: how do we choose $\boldsymbol{w}$ to minimize cost (error)

$$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

where $\hat{f}(\boldsymbol{x}_n) = \boldsymbol{w}^T\boldsymbol{x}_n$

So we want to minimize $E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$

How to fit our model to the training data?

Equivalently: how do we choose $\boldsymbol{w}$ to minimize cost (error)

$$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

where $\hat{f}(\boldsymbol{x}_n) = \boldsymbol{w}^T\boldsymbol{x}_n$

So we want to minimize $E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$

…by varying $\boldsymbol{w}$

How to fit our model to the training data?

Take the derivative with respect to $\boldsymbol{w}$, set it to zero, and solve for $\boldsymbol{w}$

p = number of predictors
N = number of data points

How to fit our model to the training data?

Take the derivative with respect to $\boldsymbol{w}$, set it to zero, and solve for $\boldsymbol{w}$

$$\nabla_w E_{in}(\boldsymbol{w}) = \nabla_w \left(\frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2 \right)$$

p = number of predictors
N = number of data points

How to fit our model to the training data?

Take the derivative with respect to $\boldsymbol{w}$, set it to zero, and solve for $\boldsymbol{w}$

$$\nabla_w E_{in}(\boldsymbol{w}) = \nabla_w \left(\frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2 \right)$$

p = number of predictors
N = number of data points

$$\nabla_w E_{in}(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial E_{in}}{\partial w_0} \\ \frac{\partial E_{in}}{\partial w_1} \\ \vdots \\ \frac{\partial E_{in}}{\partial w_p} \end{bmatrix} = \mathbf{0}$$

How to fit our model to the training data?

Take the derivative with respect to $\boldsymbol{w}$, set it to zero, and solve for $\boldsymbol{w}$

$$\nabla_w E_{in}(\boldsymbol{w}) = \nabla_w \left(\frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2 \right)$$

$$\nabla_w E_{in}(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial E_{in}}{\partial w_0} \\ \frac{\partial E_{in}}{\partial w_1} \\ \vdots \\ \frac{\partial E_{in}}{\partial w_p} \end{bmatrix} = \boldsymbol{0}$$

p = number of predictors
N = number of data points

Here we walk through the **ordinary least squares** (OLS) closed-form solution.

How to fit our model to the training data?

Take the derivative with respect to $\boldsymbol{w}$, set it to zero, and solve for $\boldsymbol{w}$

$$\nabla_w E_{in}(\boldsymbol{w}) = \nabla_w \left(\frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2 \right)$$

$$\nabla_w E_{in}(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial E_{in}}{\partial w_0} \\ \frac{\partial E_{in}}{\partial w_1} \\ \vdots \\ \frac{\partial E_{in}}{\partial w_p} \end{bmatrix} = \boldsymbol{0}$$

p = number of predictors
N = number of data points

Here we walk through the **ordinary least squares** (OLS) closed-form solution.

Could have used an iterative approach like **gradient descent**

How to fit our model to the training data?

We can rewrite our objective function:

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$$

p = number of predictors
N = number of data points

How to fit our model to the training data?

We can rewrite our objective function:

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2$$

Scalar

$$\mathbf{w}^T \in \mathbb{R}^{1\times p+1}$$

$$\mathbf{x}_n \in \mathbb{R}^{p+1\times 1}$$

p = number of predictors
N = number of data points

How to fit our model to the training data?

We can rewrite our objective function:

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\underbrace{\boldsymbol{w}^T\boldsymbol{x}_n}_{\text{Scalar}} - y_n)^2$$

$$\boldsymbol{w}^T \in \mathbb{R}^{1\times p+1}$$

$$\boldsymbol{x}_n \in \mathbb{R}^{p+1\times 1}$$

Convenient definitions:

$$\boldsymbol{y} \in \mathbb{R}^{N\times 1}$$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} \in \mathbb{R}^{N\times p+1}$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1\times 1}$$

p = number of predictors
N = number of data points

How to fit our model to the training data?

We can rewrite our objective function:

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^T\boldsymbol{x}_n - y_n)^2$$

Scalar

$$\boldsymbol{w}^T \in \mathbb{R}^{1\times p+1}$$

$$\boldsymbol{x}_n \in \mathbb{R}^{p+1\times 1}$$

We can rewrite our objective function:

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

Convenient definitions:

$$\boldsymbol{y} \in \mathbb{R}^{N\times 1}$$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix} \in \mathbb{R}^{N\times p+1}$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1\times 1}$$

p = number of predictors
N = number of data points

How to fit our model to the training data?

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

How to fit our model to the training data?

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

$$\nabla_w E_{in}(\boldsymbol{w}) = \frac{2}{N}(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y}) = \boldsymbol{0}$$

How to fit our model to the training data?

$$E_{in}(\mathbf{w}) = \frac{1}{N}(\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$$

$$\nabla_w E_{in}(\mathbf{w}) = \frac{2}{N}(\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}) = \mathbf{0}$$

$$\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} = \mathbf{0}$$

How to fit our model to the training data?

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

$$\nabla_w E_{in}(\boldsymbol{w}) = \frac{2}{N}(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y}) = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$$ (normal equation)

How to fit our model to the training data?

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

$$\nabla_w E_{in}(\boldsymbol{w}) = \frac{2}{N}(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y}) = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$$ (normal equation)

$$\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

How to fit our model to the training data?

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

$$\nabla_w E_{in}(\boldsymbol{w}) = \frac{2}{N}(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y}) = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$$ (normal equation)

$$\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Pseudoinverse $\mathbf{X}^\dagger = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$

How to fit our model to the training data?

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

$$\nabla_w E_{in}(\boldsymbol{w}) = \frac{2}{N}(\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y}) = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{0}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$$ (normal equation)

$$\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Pseudoinverse $\mathbf{X}^\dagger = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$

$$\boxed{\boldsymbol{w}^* = \mathbf{X}^\dagger\boldsymbol{y}}$$

What is the pseudoinverse?

$$\begin{array}{cc} & \text{Features} \\ \text{Samples} & \begin{bmatrix} N \times p \end{bmatrix} \end{array} \begin{bmatrix} p \times 1 \end{bmatrix} = \begin{bmatrix} N \times 1 \end{bmatrix}$$

$$\boldsymbol{X} \quad \boldsymbol{w} = \boldsymbol{y}$$

What is the pseudoinverse?

$$\begin{array}{cc} & \text{Features} \\ \text{Samples} & \begin{bmatrix} \\ N \times p \\ \\ \end{bmatrix} \end{array} \begin{bmatrix} \\ p \times 1 \\ \\ \end{bmatrix} = \begin{bmatrix} N \times 1 \\ \\ \\ \end{bmatrix}$$

$$\boldsymbol{X} \quad \boldsymbol{w} = \boldsymbol{y}$$

If $N = p$, then there are the same number of features as samples

What is the pseudoinverse?

$$\begin{array}{cc} & \text{Features} \\ \text{Samples} & \begin{bmatrix} \\ N \times p \\ \\ \end{bmatrix} \end{array} \begin{bmatrix} \\ p \times 1 \\ \\ \end{bmatrix} = \begin{bmatrix} N \times 1 \\ \\ \\ \end{bmatrix}$$

$$\boldsymbol{X} \quad \boldsymbol{w} = \boldsymbol{y}$$

If $N = p$, then there are the same number of features as samples

If $N > p$, then the system of equations is overdetermined (more samples than features)

What is the pseudoinverse?

Consider the case when $N = 3, p = 2$

What is the pseudoinverse?

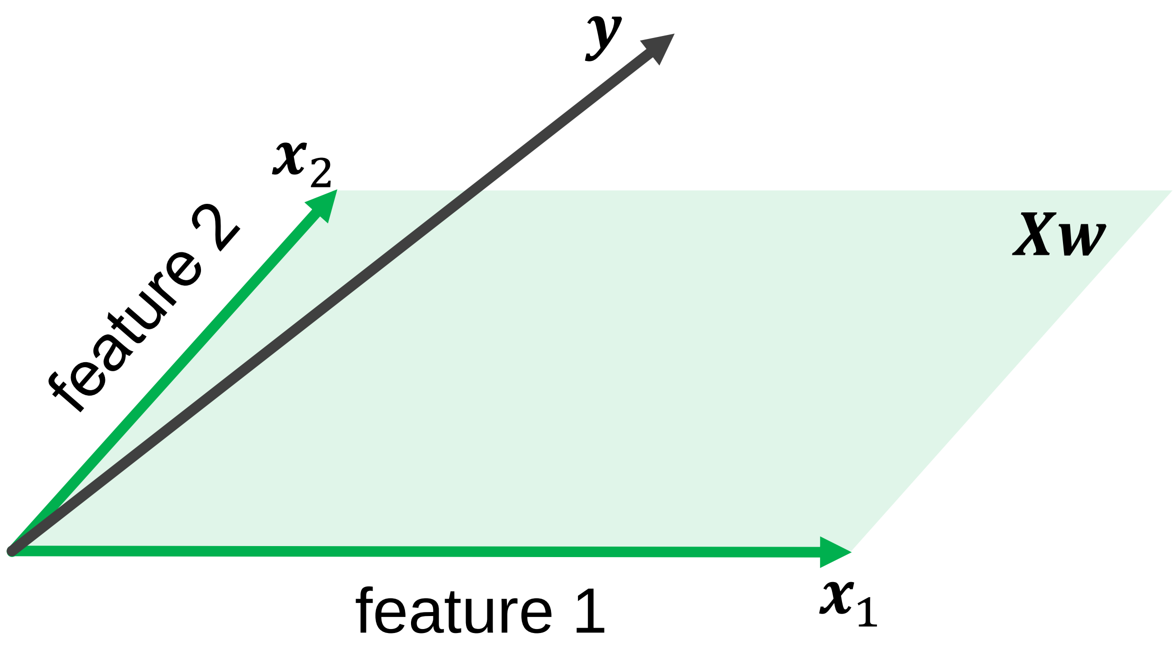
Consider the case when $N = 3$, $p = 2$

$$\begin{array}{cc} & \text{Features} \\ \text{Samples} & \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} \end{array} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

What is the pseudoinverse?

Consider the case when $N = 3$, $p = 2$

$$\begin{array}{cc} & \text{Features} \\ \text{Samples} & \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} \end{array} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

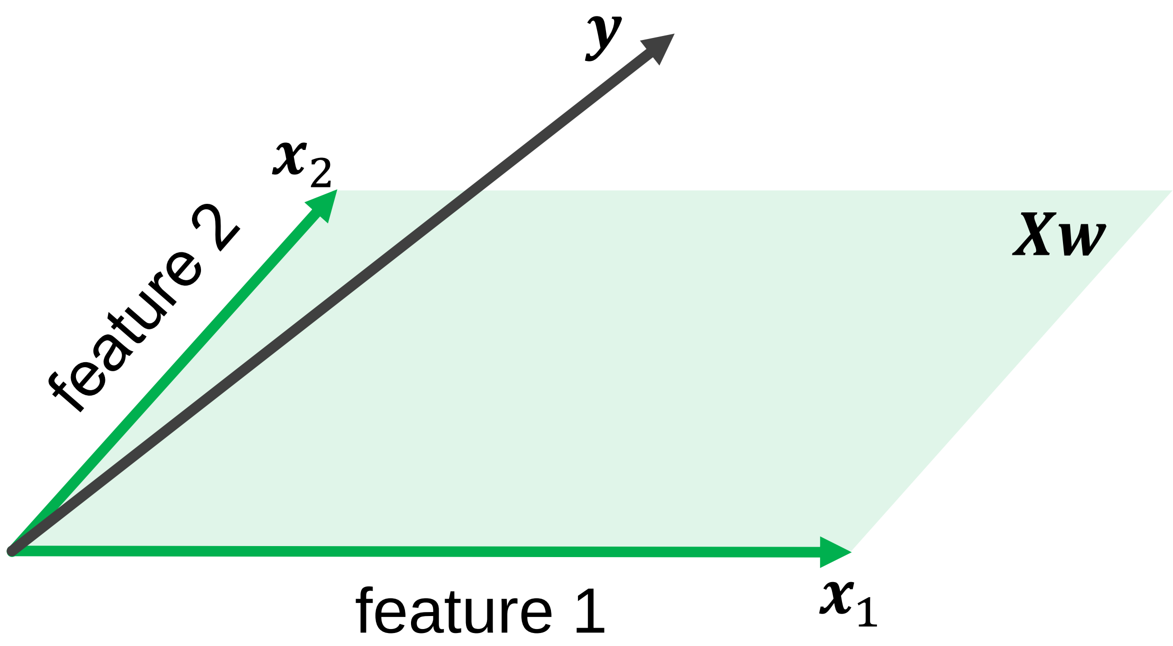


What is the pseudoinverse?

Consider the case when $N = 3$, $p = 2$

$$
\begin{array}{cc}
 & \text{Features} \\
\text{Samples} & \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix}
\end{array}
\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}
$$

$$
\boldsymbol{X} \quad \boldsymbol{w} \neq \boldsymbol{y}
$$



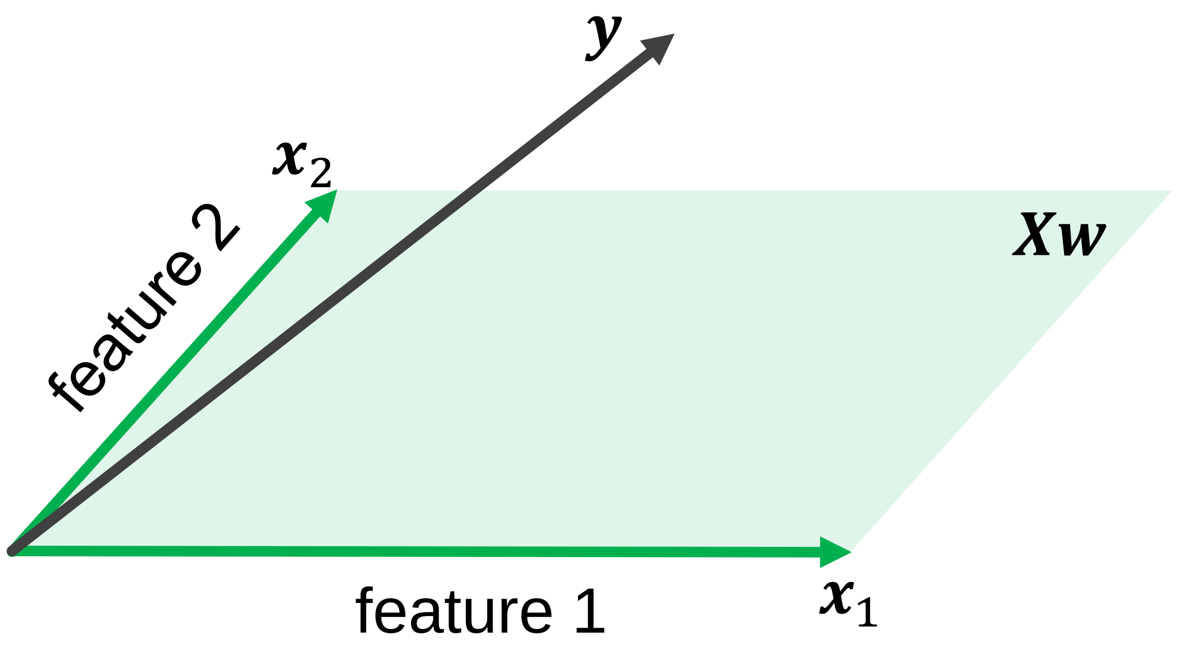
What is the pseudoinverse?

Consider the case when $N = 3, p = 2$

$$\begin{array}{cc} & \text{Features} \\ \text{Samples} & \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} \end{array} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\boldsymbol{X} \quad \boldsymbol{w} \neq \boldsymbol{y}$$

We CAN solve for the least squares solution: $\boldsymbol{X}^{\dagger}\boldsymbol{w}^{*} = \boldsymbol{y}$



What is the pseudoinverse?

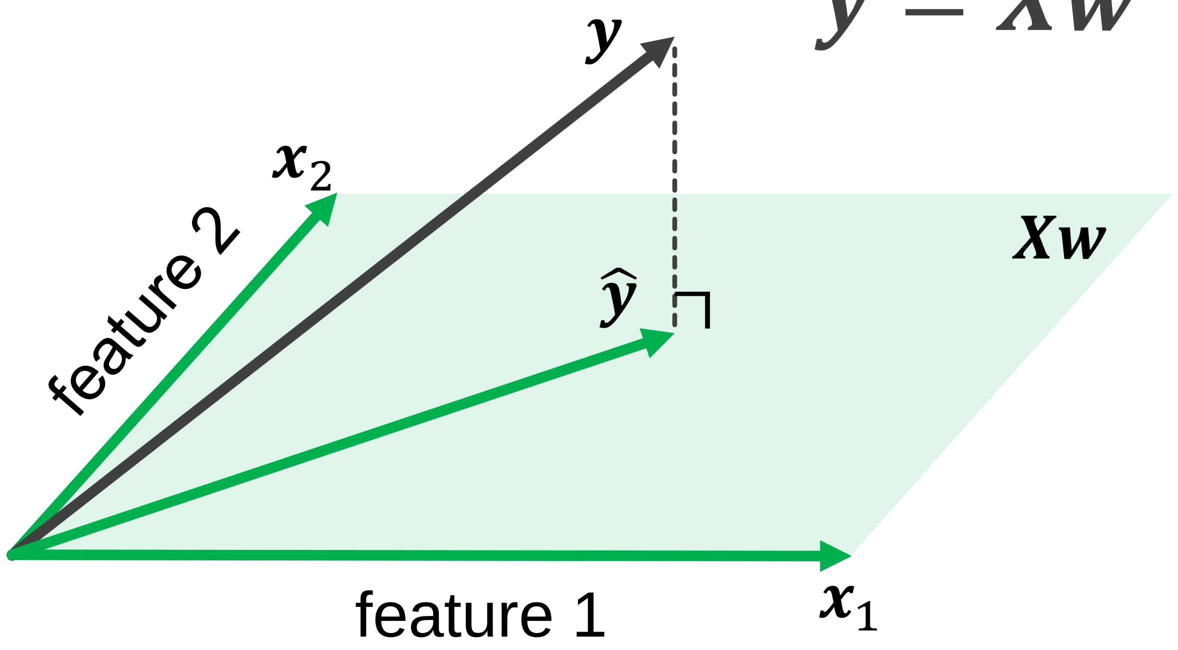
Consider the case when $N = 3$, $p = 2$

$$
\begin{array}{cc}
 & \text{Features} \\
\text{Samples} & \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix}
\end{array}
\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}
$$

$$\boldsymbol{X} \quad \boldsymbol{w} \neq \boldsymbol{y}$$

We CAN solve for the least squares solution: $\boldsymbol{X}^{\dagger}\boldsymbol{w}^{*} = \boldsymbol{y}$

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w}^{*}$$



What is the pseudoinverse?

Consider the case when $N = 3$, $p = 2$

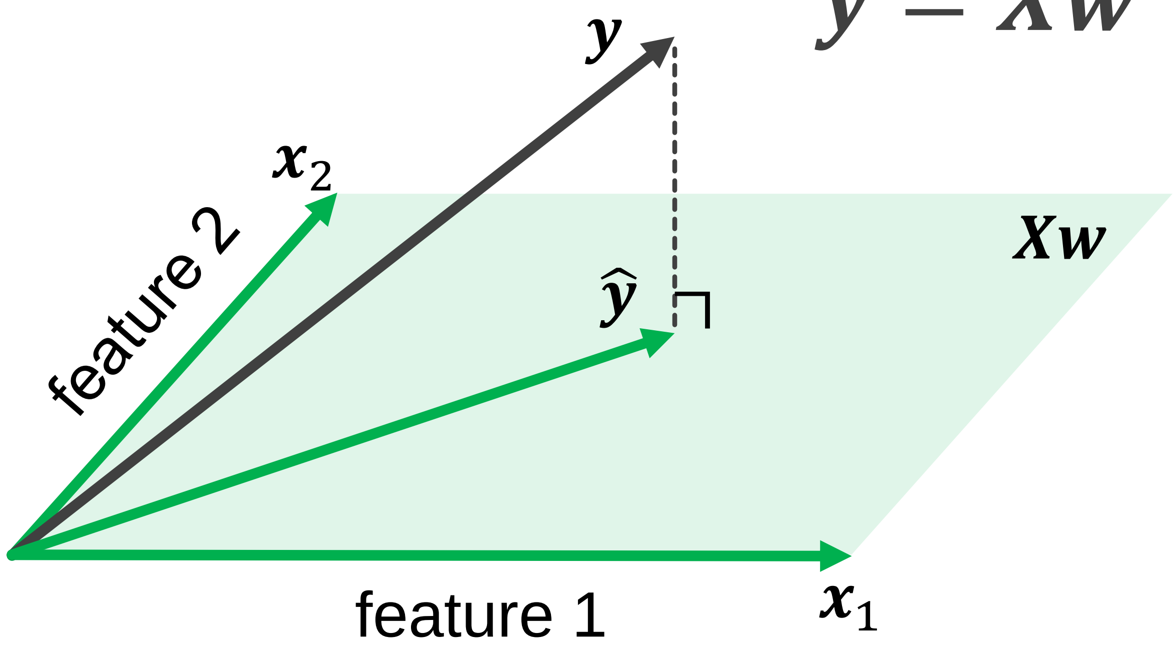
$$\begin{array}{cc} & \text{Features} \\ \text{Samples} & \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} \end{array} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\boldsymbol{X} \quad \boldsymbol{w} \neq \boldsymbol{y}$$

We CAN solve for the least squares solution: $\boldsymbol{X}^{\dagger}\boldsymbol{w}^{*} = \boldsymbol{y}$

The least squares solution is the best we can do given $N > p$

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w}^{*}$$



Common paradigm for model fitting

Common paradigm for model fitting

1. Choose a **hypothesis set of models** to train
   (e.g. linear regression with 4 predictor variables)

Common paradigm for model fitting

1. Choose a **hypothesis set of models** to train
(e.g. linear regression with 4 predictor variables)

2. Identify a **cost function** to measure the model fit to the training data
(e.g. mean square error)

Common paradigm for model fitting

1. Choose a **hypothesis set of models** to train
   (e.g. linear regression with 4 predictor variables)

2. Identify a **cost function** to measure the model fit to the training data
   (e.g. mean square error)

3. **Optimize** model **parameters** to minimize cost
   (e.g. closed form solution using the normal equations for OLS)

Much of machine learning is optimizing a cost function

What about classification?

Moving from regression to classification

Regression

$$y = \sum_{i=0}^{N} w_i x_i$$

Classification
(perceptron model)

$$y = \left(\sum_{i=0}^{N} w_i x_i \right)$$

Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

Regression

$$y = \sum_{i=0}^{N} w_i x_i$$

Classification
(perceptron model)

$$y = sign\left(\sum_{i=0}^{N} w_i x_i\right)$$

Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

Regression

$$y = \sum_{i=0}^{N} w_i x_i$$

Classification
(perceptron model)

$$y = sign\left(\sum_{i=0}^{N} w_i x_i\right)$$

where

$$sign(x) = \begin{cases} 1 & x > 0 \\ -1 & else \end{cases}$$

Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

Regression

$$y = \sum_{i=0}^{N} w_i x_i$$

Classification
(perceptron model)

$$y = sign\left(\sum_{i=0}^{N} w_i x_i\right)$$

$$y = \begin{cases} 1 & \sum_{i=0}^{N} w_i x_i > 0 \\ -1 & else \end{cases}$$

where

$$sign(x) = \begin{cases} 1 & x > 0 \\ -1 & else \end{cases}$$

Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

Linear Regression

$$\hat{f}(\boldsymbol{x}) = \sum_{i=0}^{N} w_i x_i$$

Linear Classification

(perceptron)

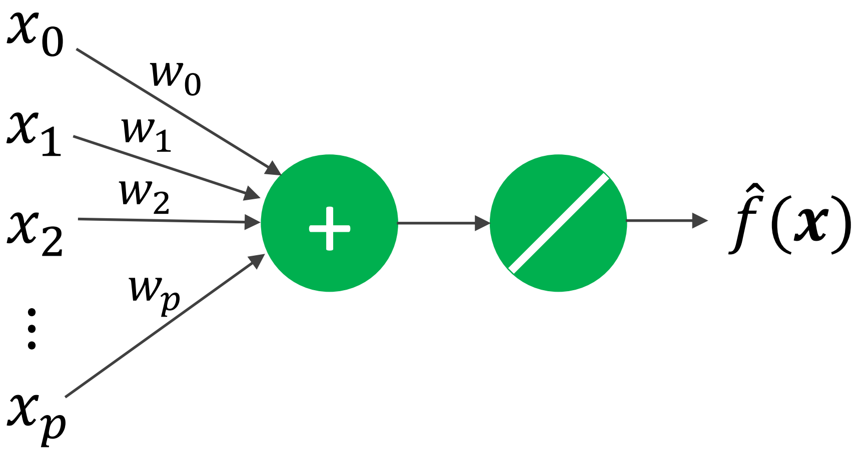
$$\hat{f}(\boldsymbol{x}) = sign\left(\sum_{i=0}^{N} w_i x_i\right)$$

Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

Linear Regression

$$\hat{f}(\boldsymbol{x}) = \sum_{i=0}^{N} w_i x_i$$



Linear Classification
(perceptron)

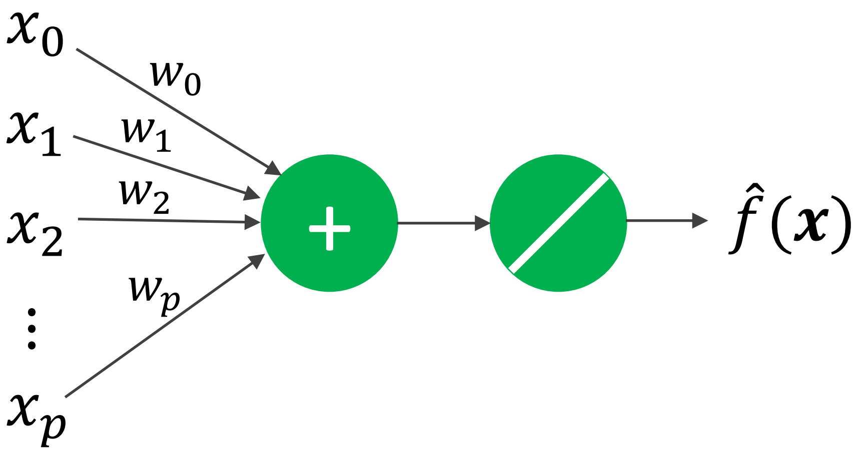
$$\hat{f}(\boldsymbol{x}) = sign\left(\sum_{i=0}^{N} w_i x_i\right)$$

Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

Linear Regression

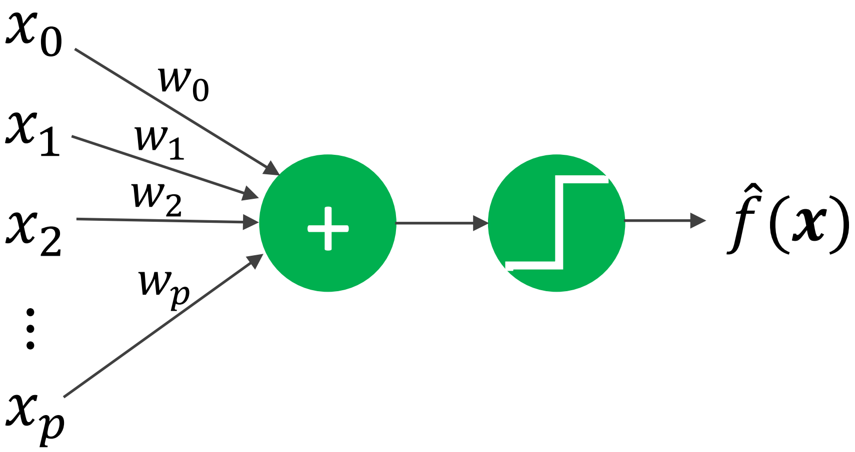
$$\hat{f}(\boldsymbol{x}) = \sum_{i=0}^{N} w_i x_i$$



Linear Classification

(perceptron)

$$\hat{f}(\boldsymbol{x}) = sign\left(\sum_{i=0}^{N} w_i x_i\right)$$

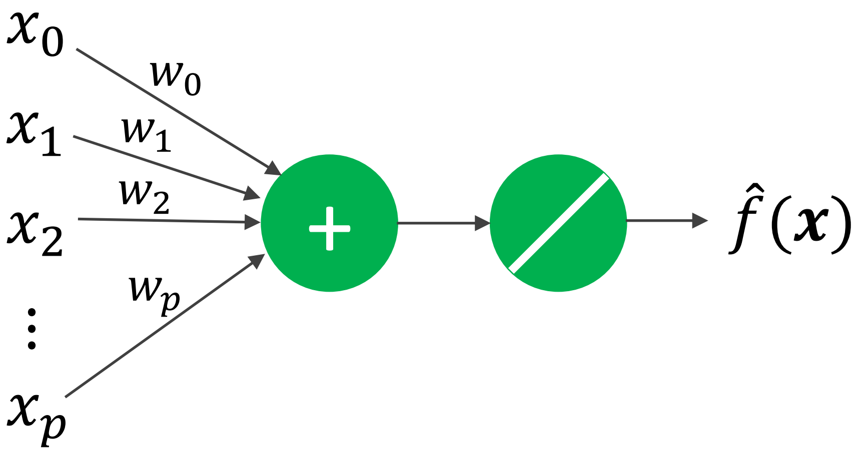


Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

Linear Regression

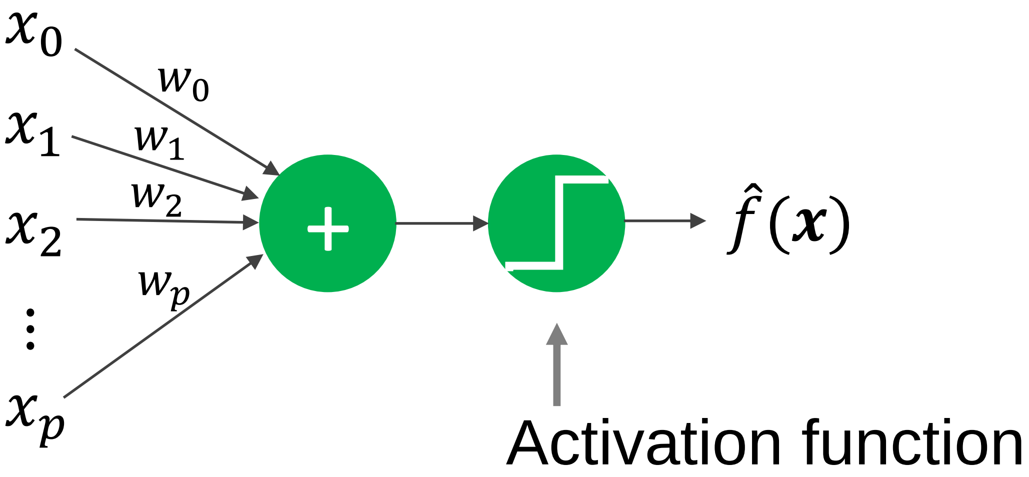
$$\hat{f}(\boldsymbol{x}) = \sum_{i=0}^{N} w_i x_i$$



Linear Classification

(perceptron)

$$\hat{f}(\boldsymbol{x}) = sign\left(\sum_{i=0}^{N} w_i x_i\right)$$



Source: Abu-Mostafa, Learning from Data, Caltech

Takeaways

Linear models are **linear in the weights**

Linear models can be used for **both regression and classification**

Model fitting/training (valid beyond linear models):

- Choose a hypothesis set of models to train
- Identify a cost function
- **Optimize the cost function** by adjusting model parameters