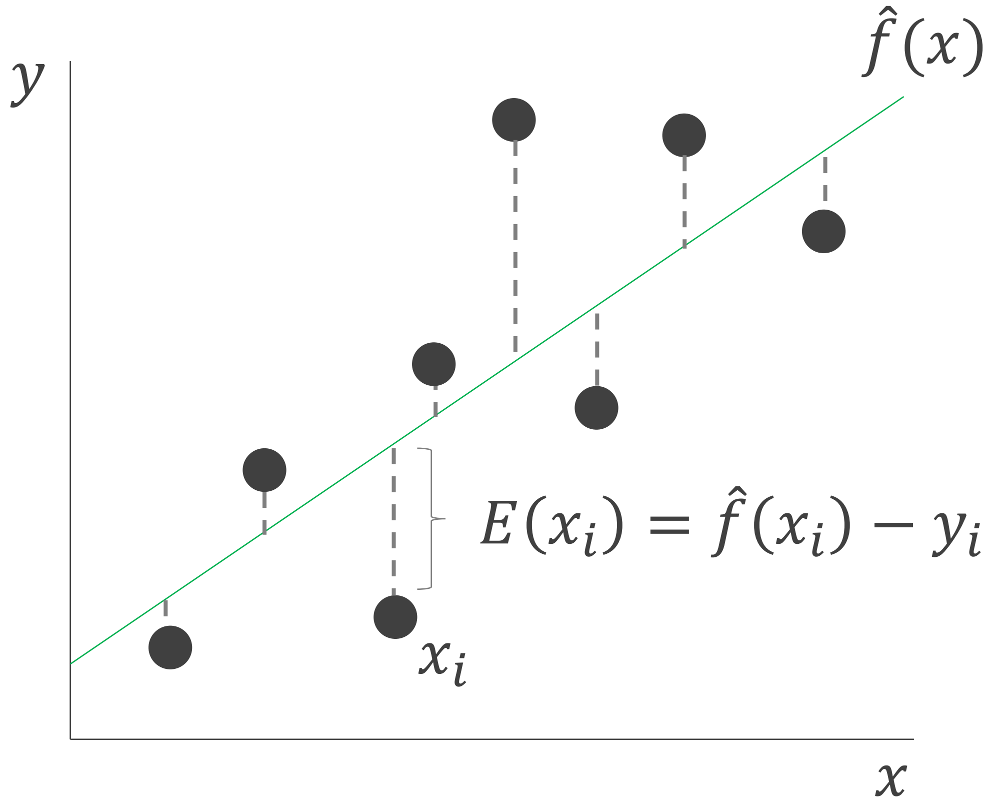
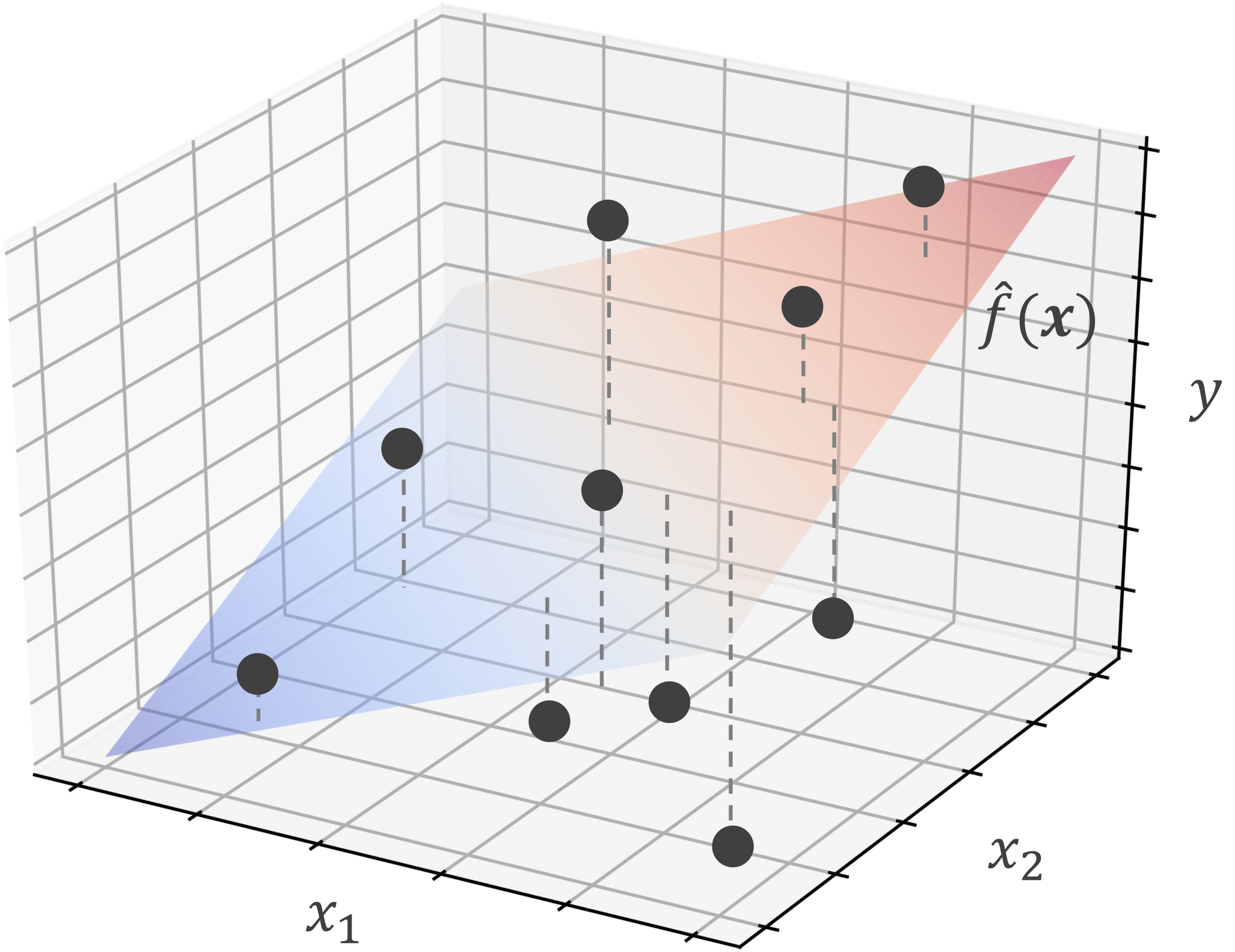
Machine Learning III

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 $\hat{y} = \hat{f}(\boldsymbol{x}) = \sum_{i=0}^{p} w_i x_i$ 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 quantify 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** (a.k.a. loss, error, or objective)

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)

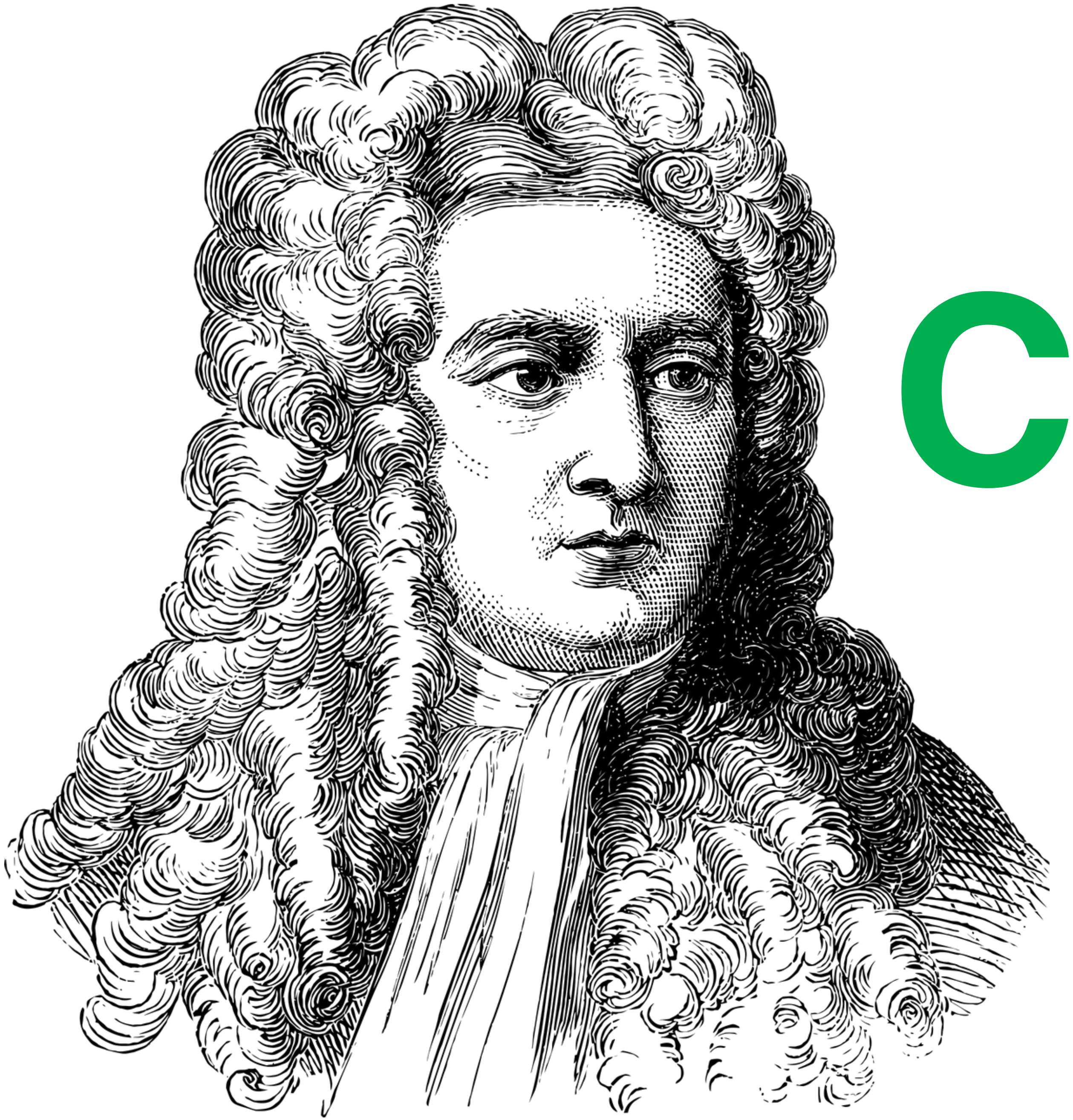
$$E_{in}(\hat{f}) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{f}(\boldsymbol{x}_n) - y_n\right)^2$$

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

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 do we do that?



Calculus

A moment of calculus

Function of one variable

$$f(x) = ax + bx^2$$

Function of multiple variables

$$f(x_1, x_2) = ax_1 + bx_2$$

Derivative

$$\frac{df}{dx} = a + 2bx$$

Partial Derivative

$$\frac{\partial f}{\partial x_1} = a$$

$$\frac{\partial f}{\partial x_2} = b$$

May also treat parameters as variables and take their partial derivative $\frac{\partial f}{\partial b} = x_2$

Gradient

$$\nabla_{\boldsymbol{x}} f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

$$= \begin{bmatrix} a \\ b \end{bmatrix}$$

How to fit our model to the training data?

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

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

p = number of predictors
N = number of data points

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

Size: $[p+1 \times 1]$ or $\mathbb{R}^{p+1\times 1}$

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

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

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

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

Classification
(perceptron model)

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

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

where

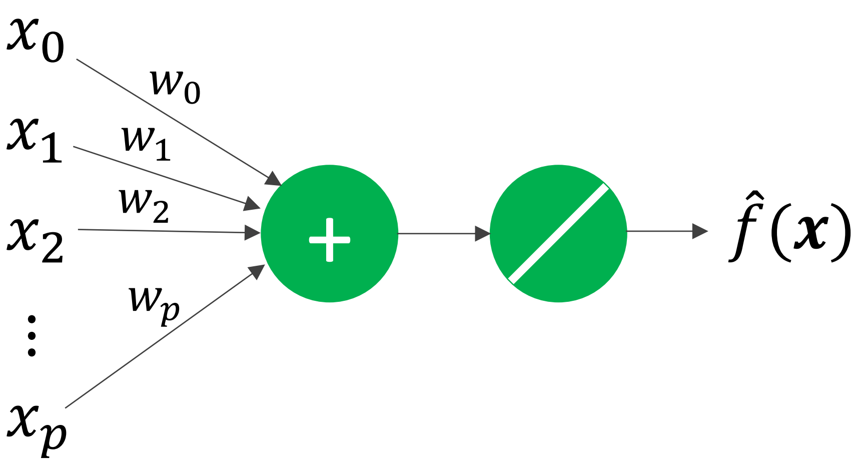
$$sign(x) = \begin{cases} 1 & x > 0 \\ -1 & else \end{cases}$$

Source: Abu-Mostafa, Learning from Data, Caltech

Moving from regression to classification

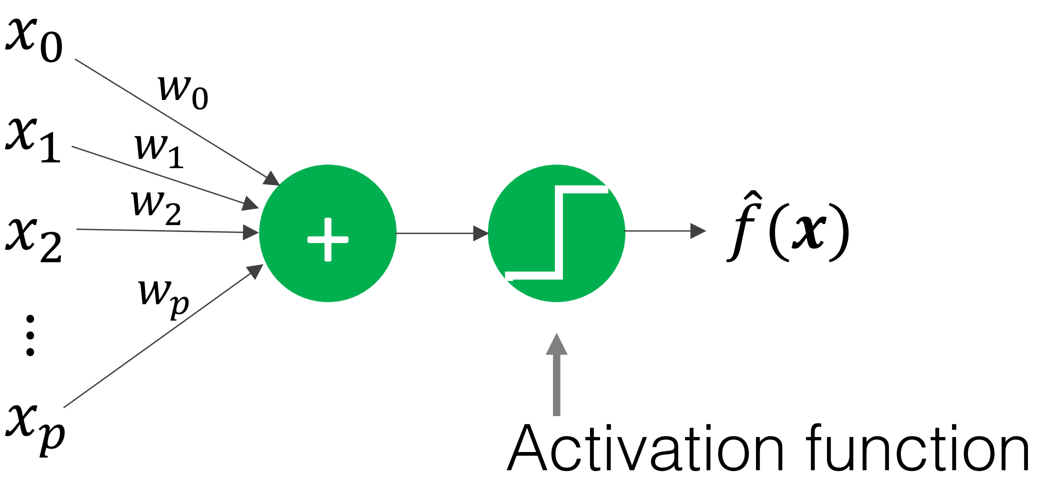
Linear Regression

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



Linear Classification
(perceptron)

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



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

How can we…

model nonlinear relationships?

use linear models for classification?

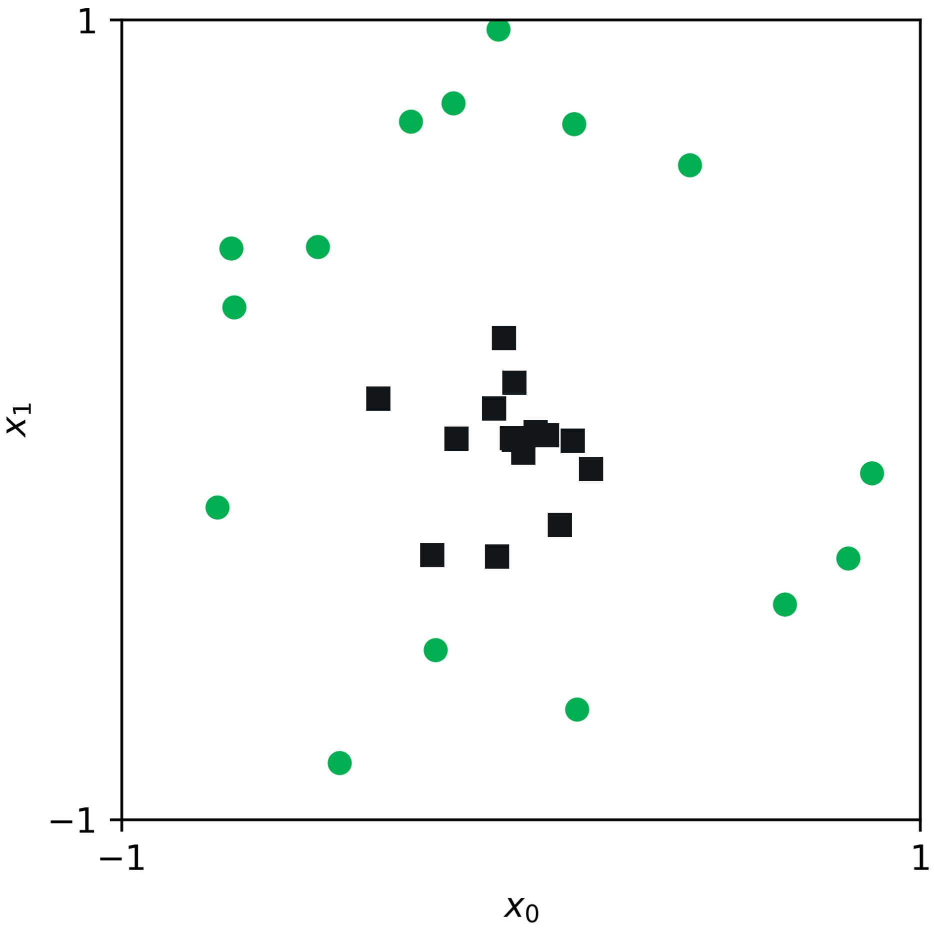
choose the parameters to fit our model to training data

Can we model nonlinear relationships?

Limitations of linear decision boundaries

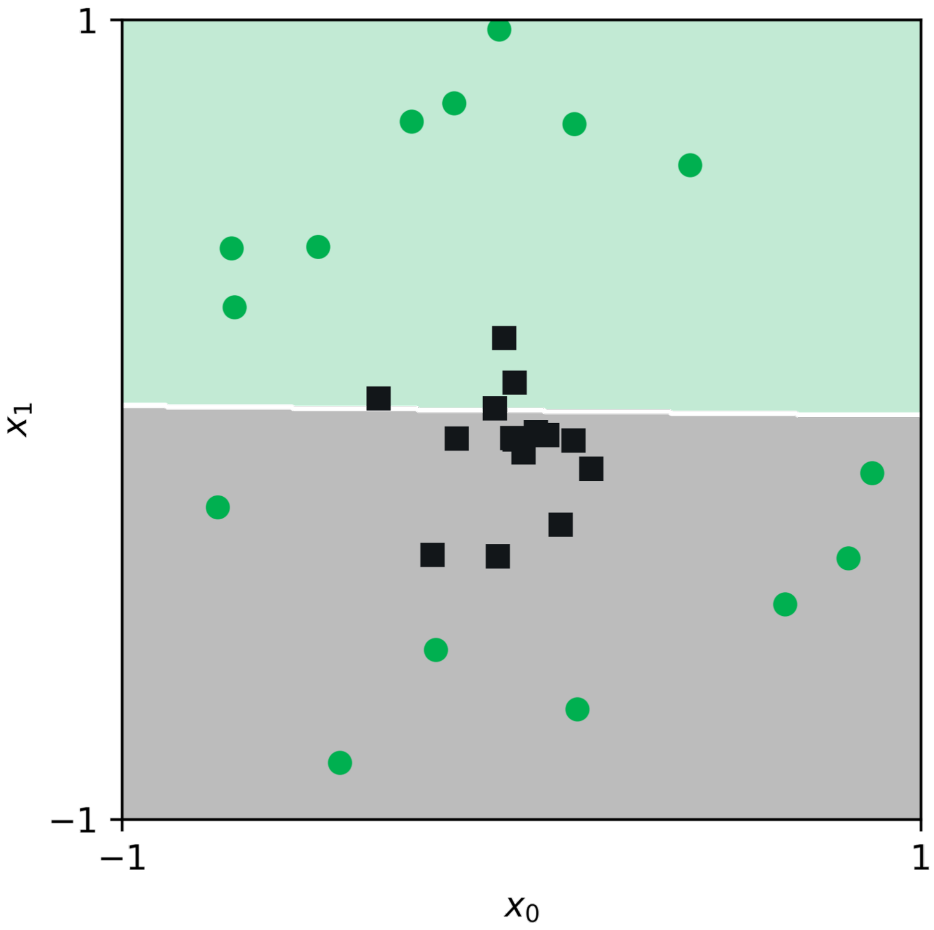
Original data

$\boldsymbol{x}$



Classify the features in this X-space

$\hat{f}_{\boldsymbol{x}}(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^T \boldsymbol{x})$



Transformations of features

Consider a digits example…

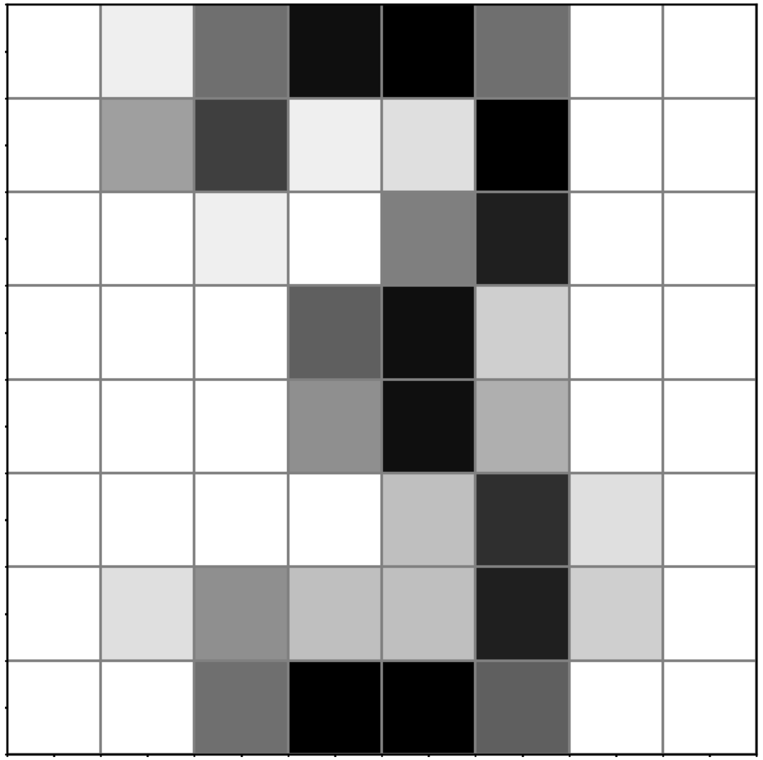
$$\boldsymbol{x} = [x_1, x_2, x_3, \dots, x_{64}]$$

We could **create features** based on the raw features. For example:

$$\boldsymbol{z} = [x_3 x_5, x_3^2, \frac{x_{64}}{x_{42}}]$$

Which can be written simply as variables in a new feature space:

$$\boldsymbol{z} = [z_1, z_2, z_3]$$

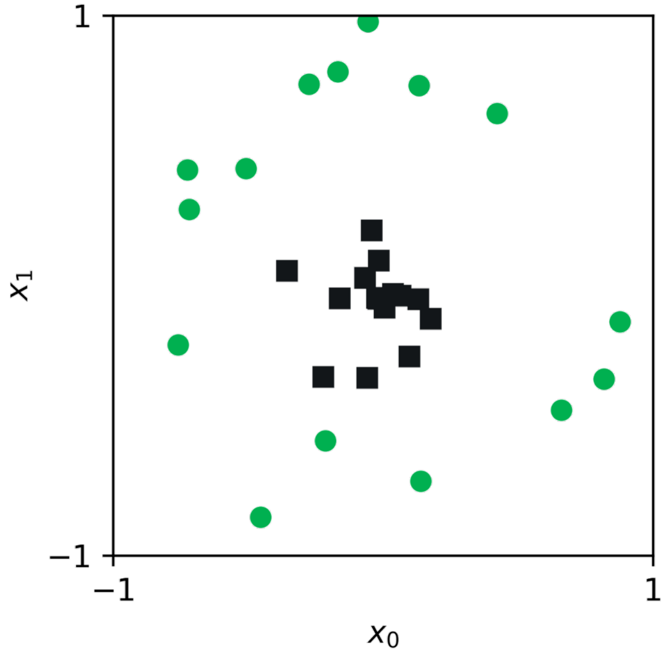


Source: Abu-Mostafa, Learning from Data, Caltech

1

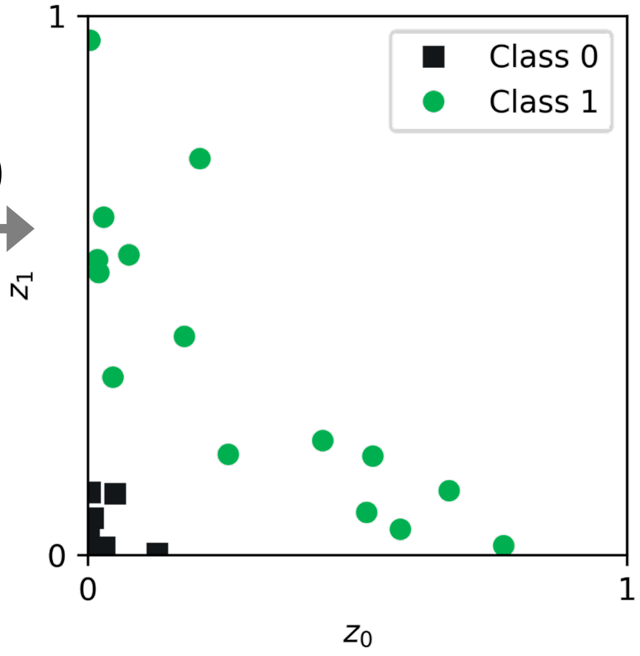
Original data

$\boldsymbol{x}$



transform the data

$\boldsymbol{z} = \Phi(\boldsymbol{x})$



2

This example transform is quadratic

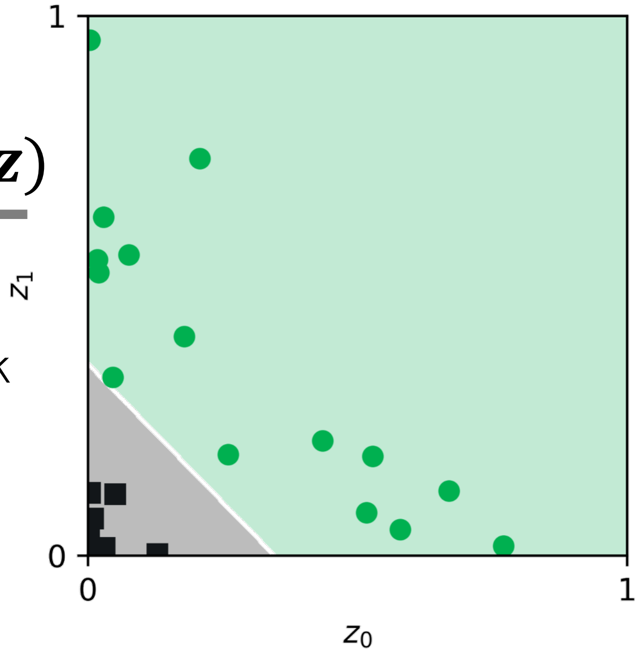
$$z_i = \Phi(x_i) = x_i^2$$

$$z_0 = x_0^2$$

$$z_1 = x_1^2$$

Classify the features in this Z-space

$$\hat{f}_z(\boldsymbol{z}) = \text{sign}(\boldsymbol{w}^T\boldsymbol{z})$$



A new **representation** of our data

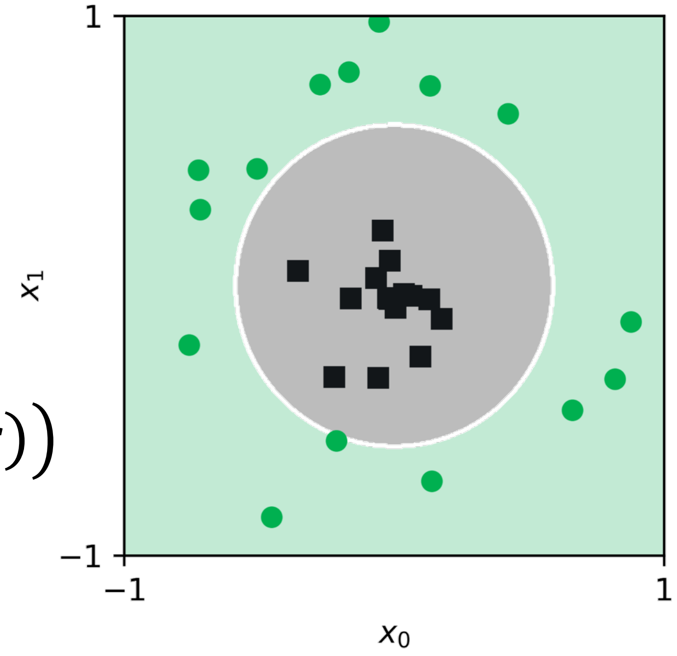
3

$\boldsymbol{x} = \Phi^{-1}(\boldsymbol{z})$

transform the data back

$$x_0 = z_0^{1/2}$$

$$x_1 = z_1^{1/2}$$



Predictions in the original X-space

$$\hat{f}(\boldsymbol{x}) = \hat{f}_z(\Phi(\boldsymbol{x}))$$

4