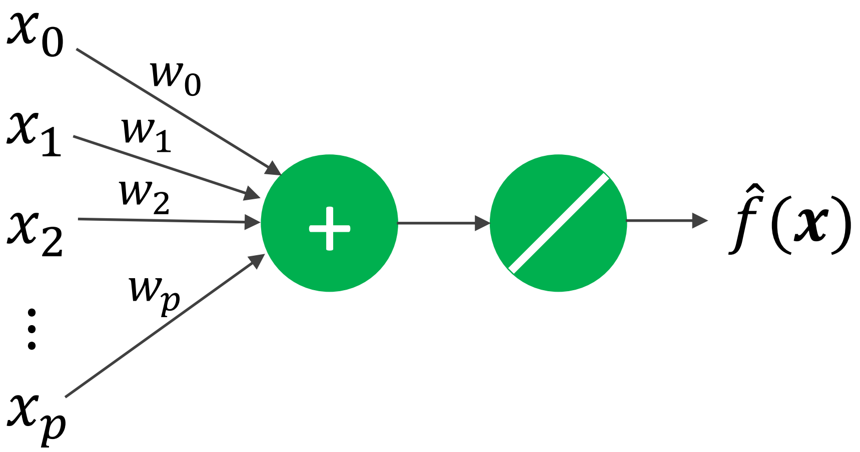
Linear models II

Lecture 05

Recap on linear models

Linear Regression

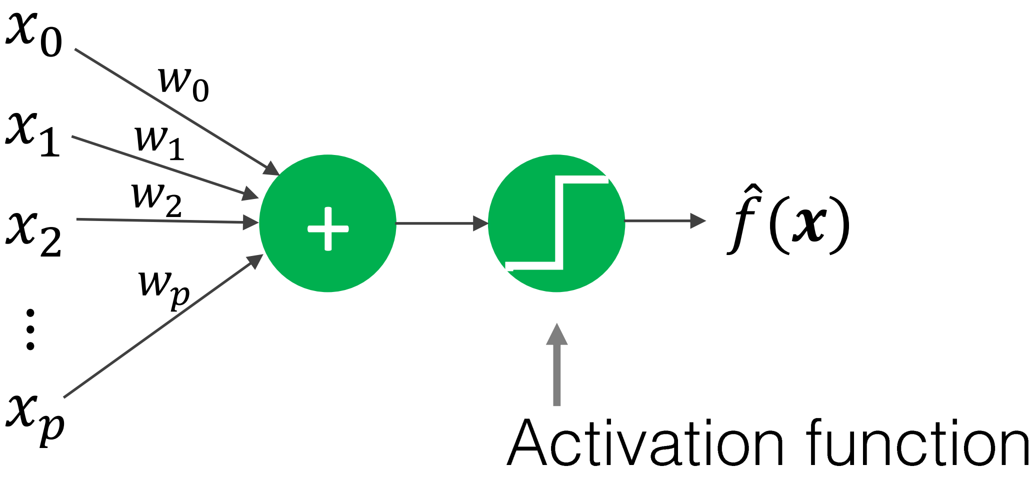
$$\hat{f}(\boldsymbol{x}) = \sum_{i=0}^{p} w_i x_i$$



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

How can we…

model nonlinear relationships?

use linear models for classification?

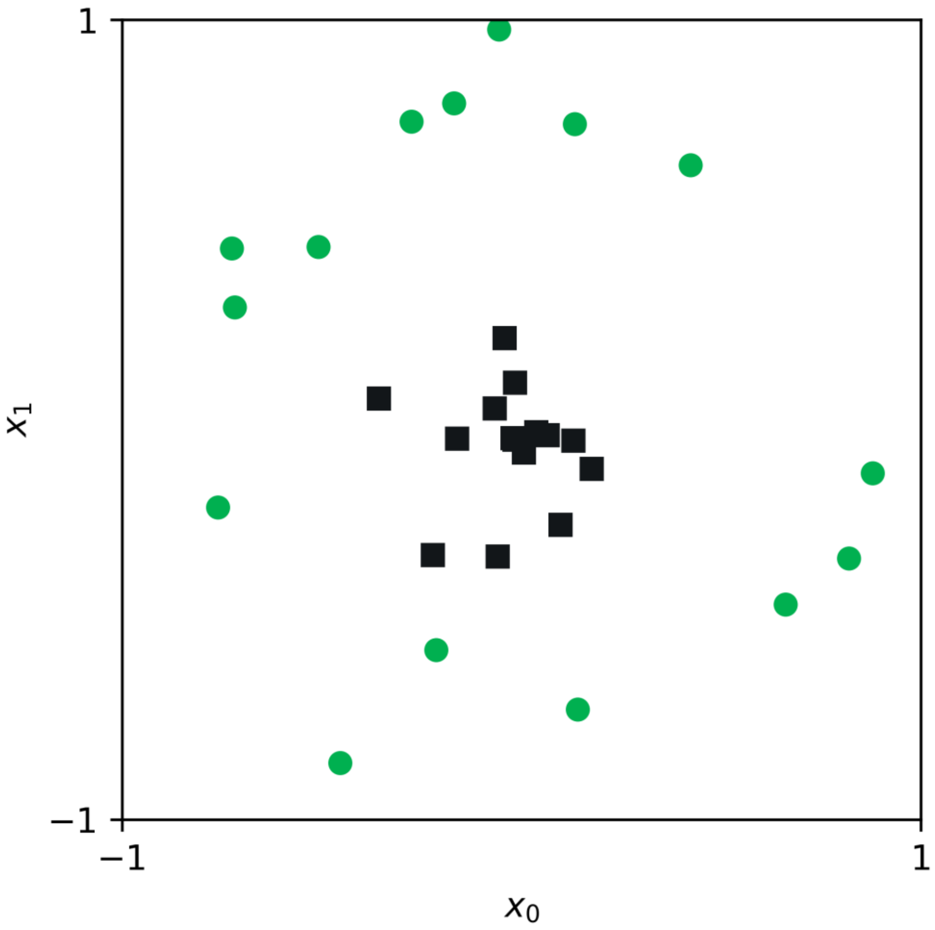
choose the parameters to fit our model to training data

Can we model nonlinear relationships?

Limitations of linear decision boundaries

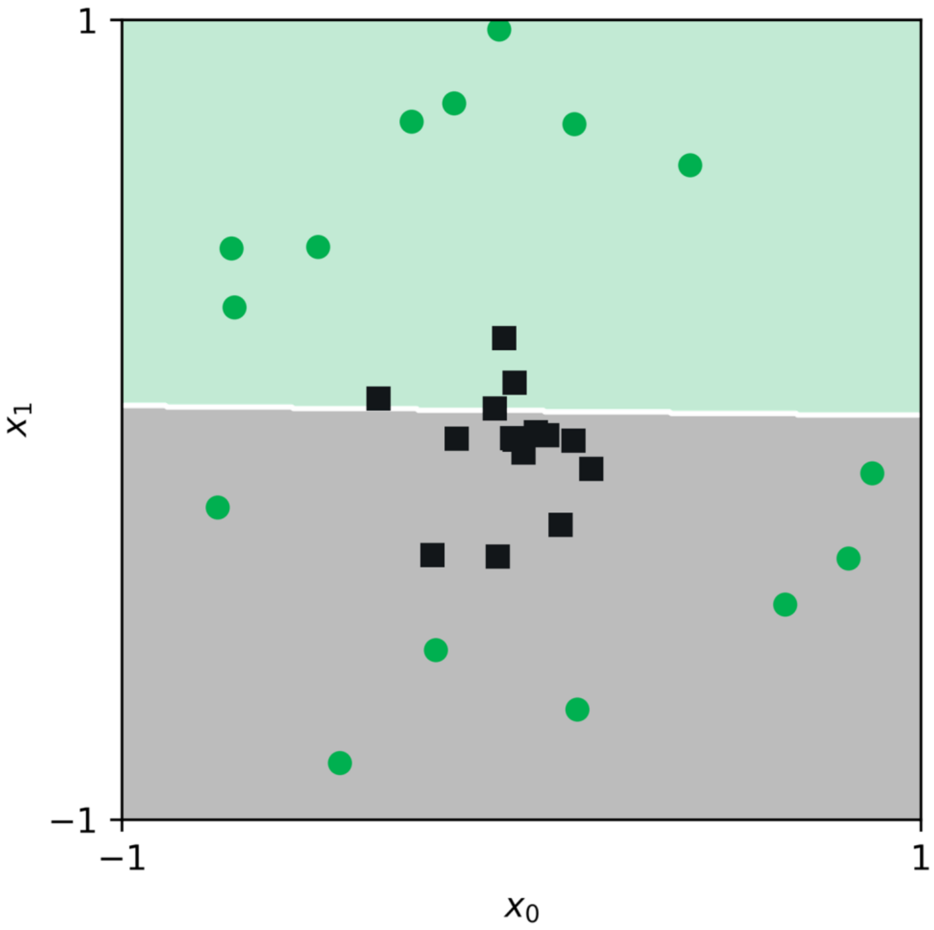
Original data

$$\boldsymbol{x}$$



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…

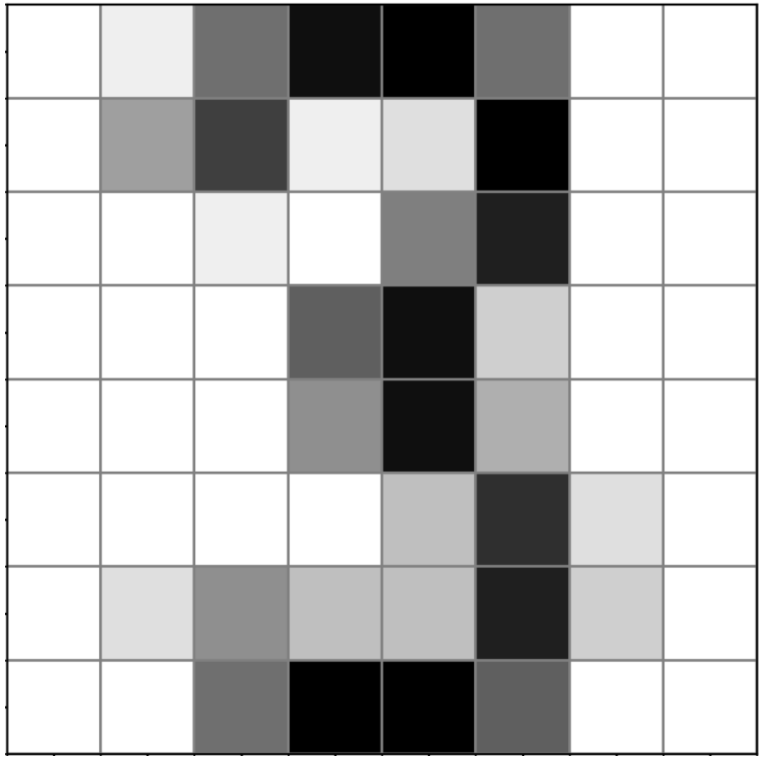
$$\boldsymbol{x} = [x_1, x_2, x_3, \dots, x_{64}]$$

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:

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

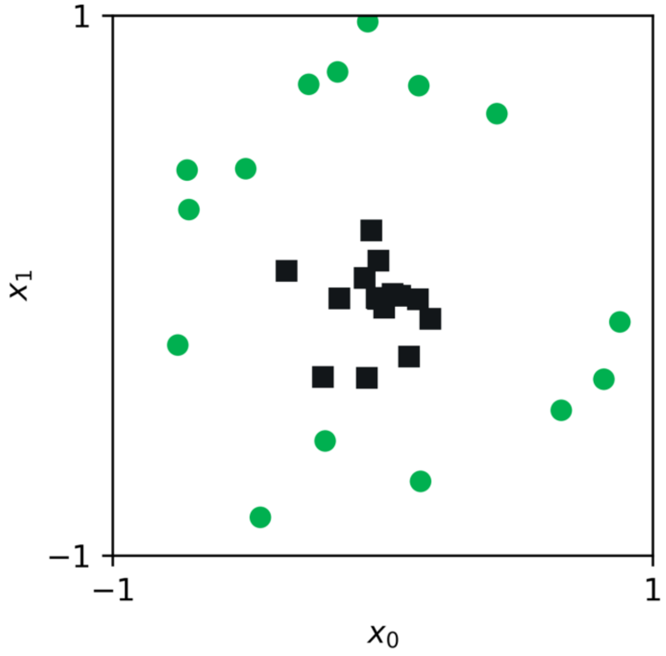


Source: Abu-Mostafa, Learning from Data, Caltech

1

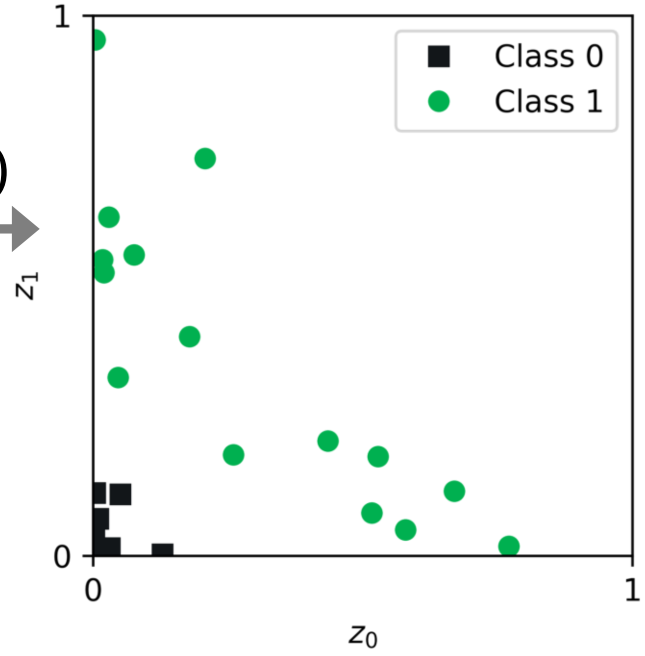
Original data

$$\boldsymbol{x}$$



transform
the data

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



2

This example transform is quadratic

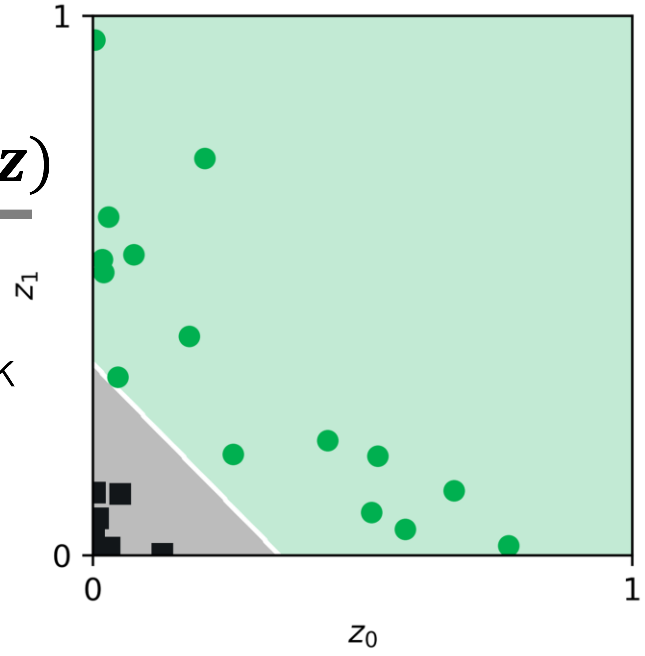
$$z_i = \Phi(x_i) = x_i^2$$

$$z_0 = x_0^2$$

$$z_1 = x_1^2$$

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

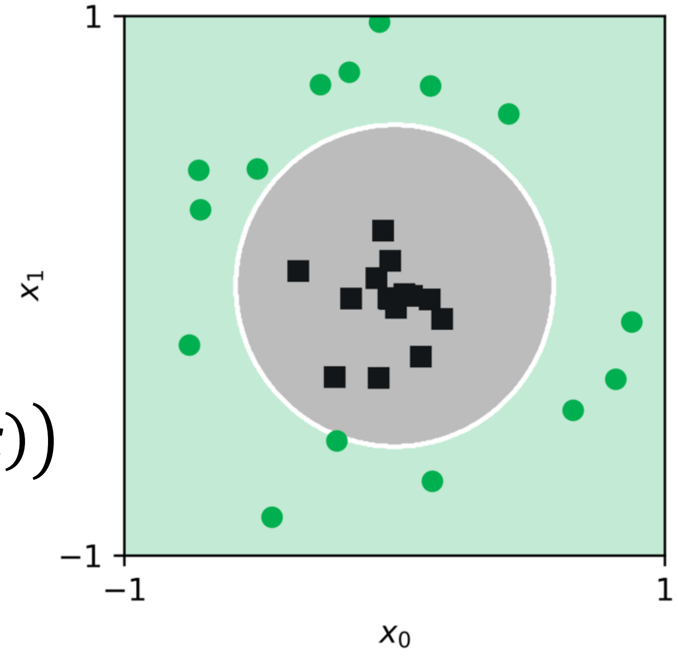
3

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

transform
the data back

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

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



Predictions in the original X-space

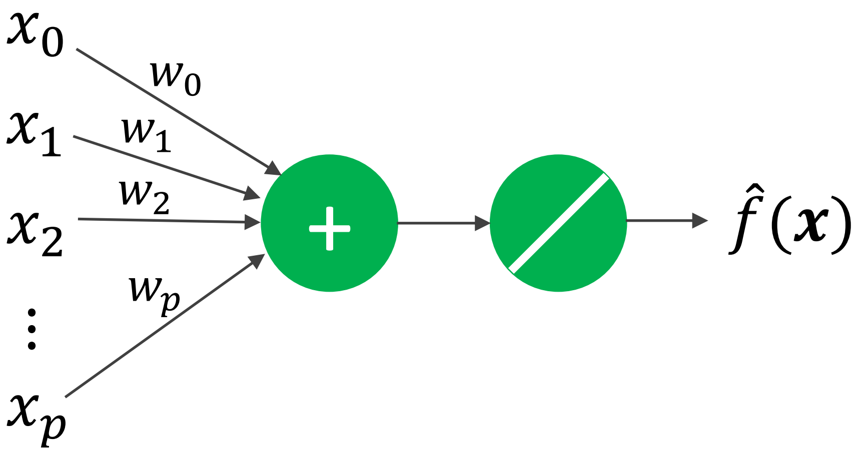
$$\hat{f}(\boldsymbol{x}) = \hat{f}_z\big(\Phi(\boldsymbol{x})\big)$$

4

Moving from regression to classification

Linear Regression

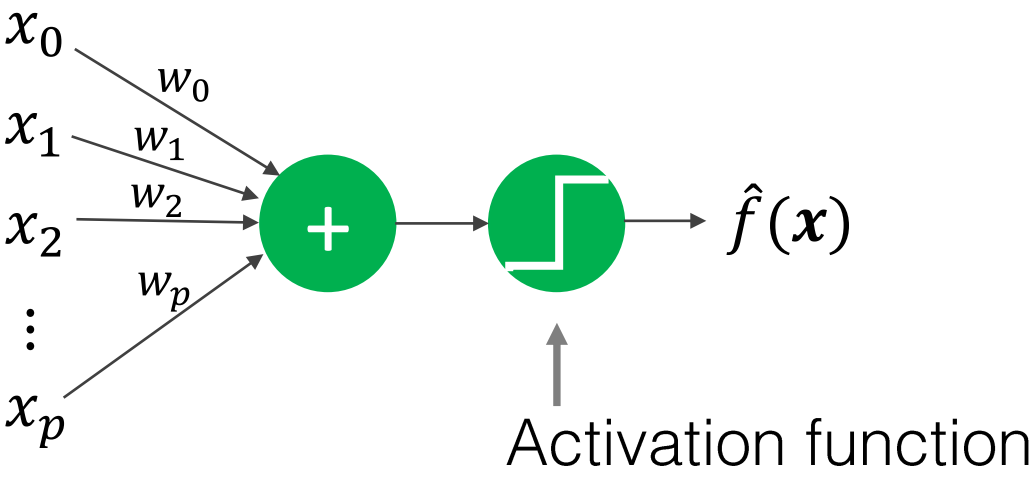
$$\hat{f}(\boldsymbol{x}) = \sum_{i=0}^{p} w_i x_i$$



Linear Classification

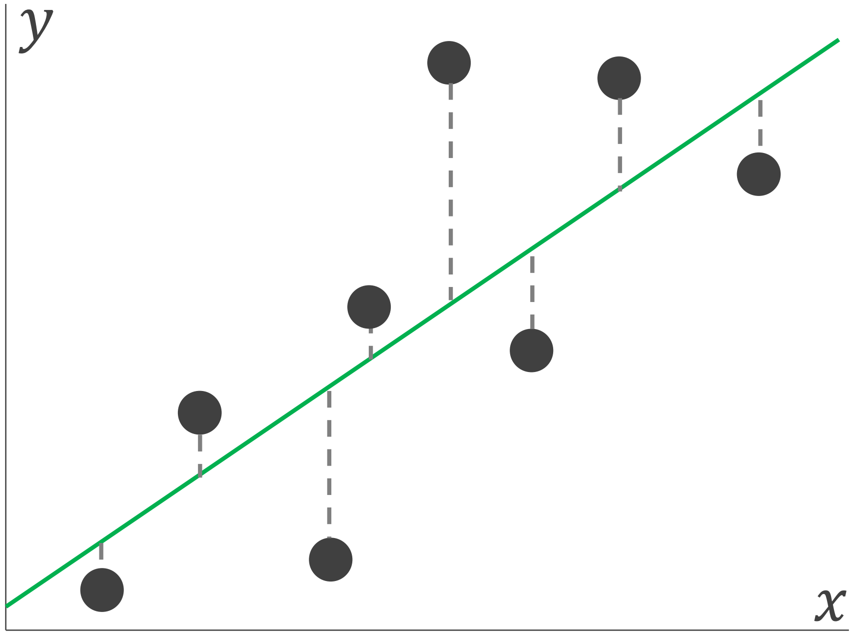
(perceptron)

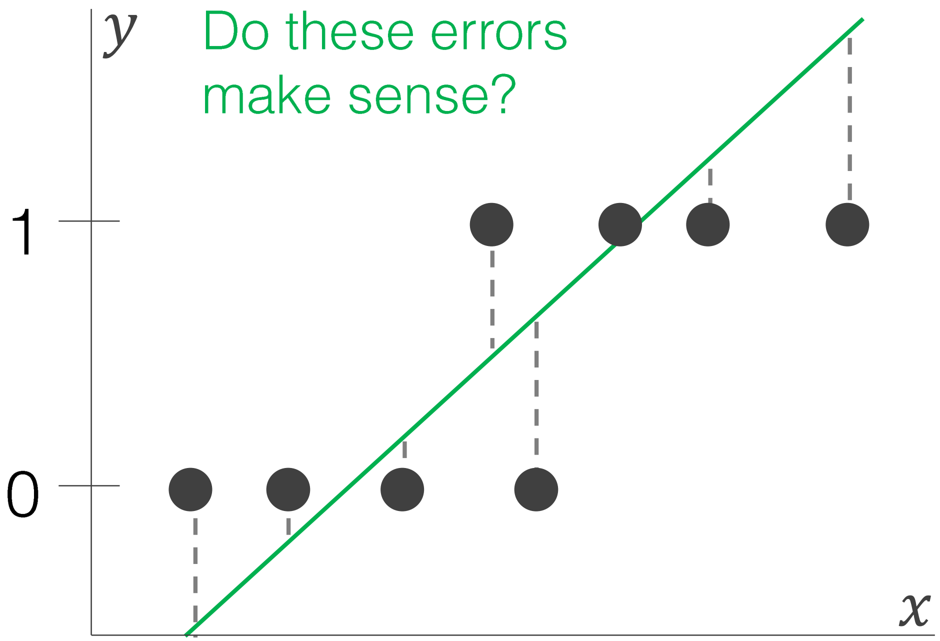
$$\hat{f}(\boldsymbol{x}) = sign\left(\sum_{i=0}^{p} w_i x_i\right)$$



Source: Abu-Mostafa, Learning from Data, Caltech

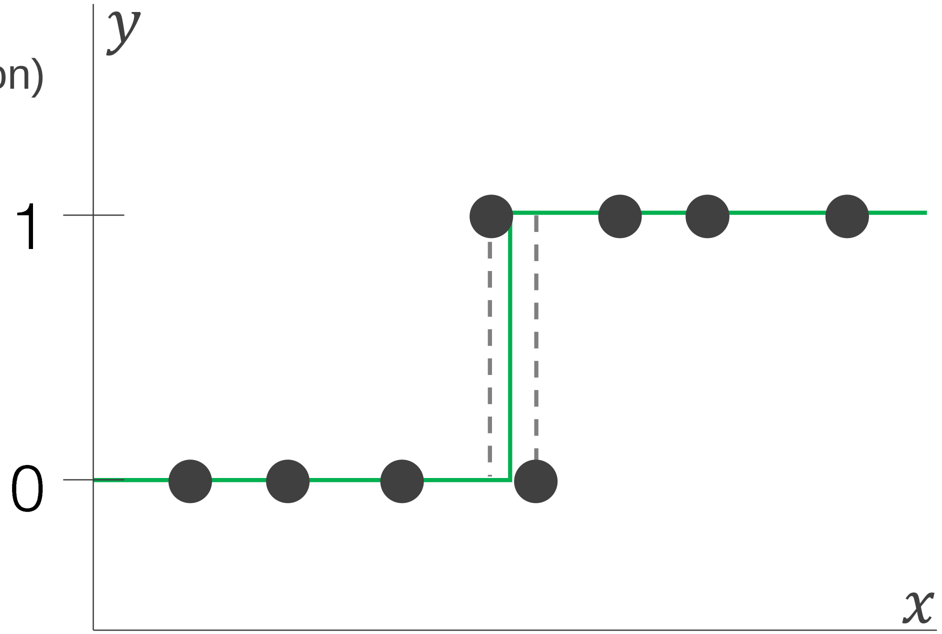
Linear regression
(linear activation)

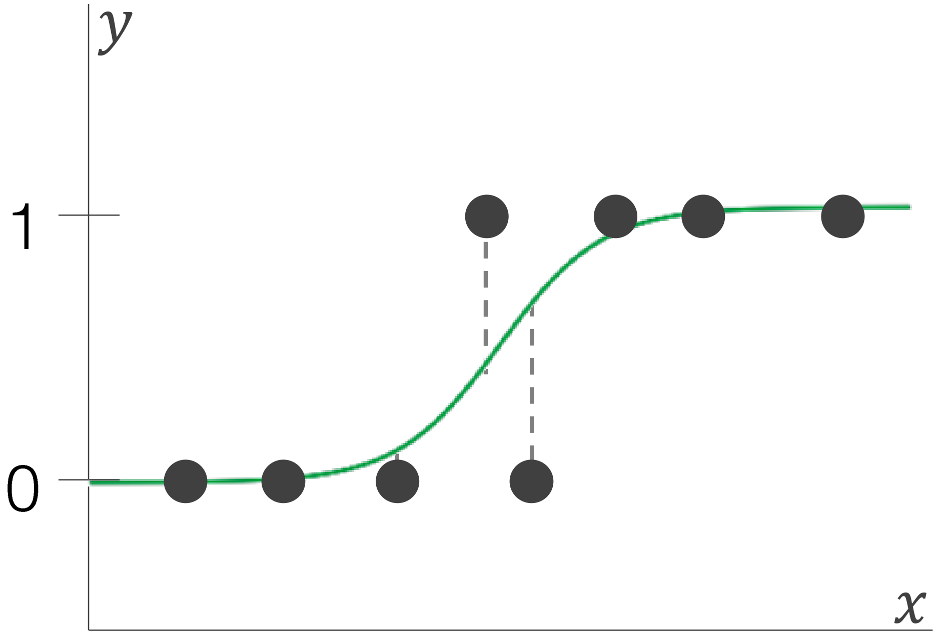




Linear regression
applied to a classification problem
(linear activation)

Perceptron
(sign activation)

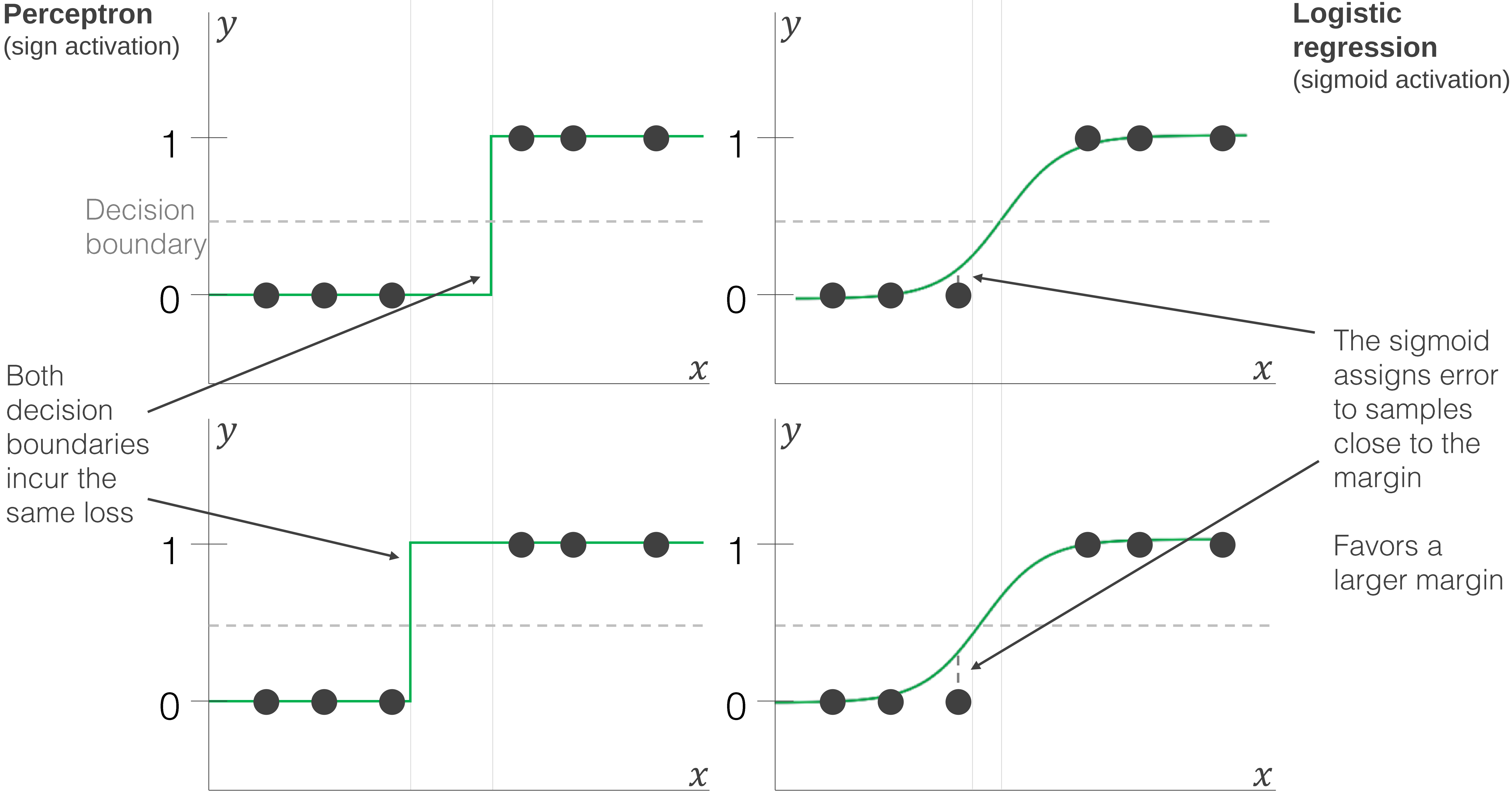




Logistic regression
(sigmoid activation)

Perceptron
(sign activation)

Logistic regression
(sigmoid activation)



Sigmoid function

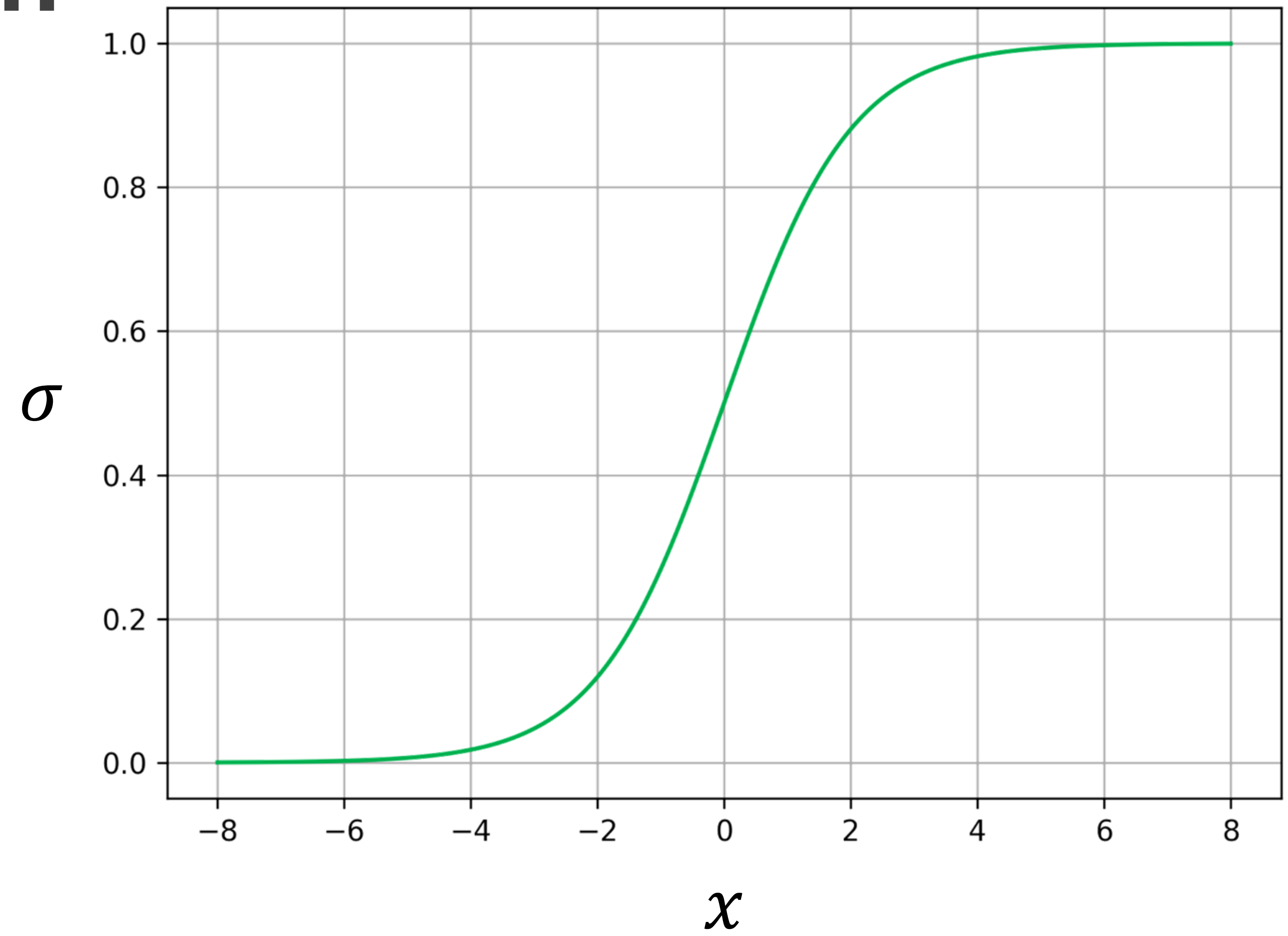
Definition

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

Useful properties

$$\sigma(-x) = 1 - \sigma(x)$$

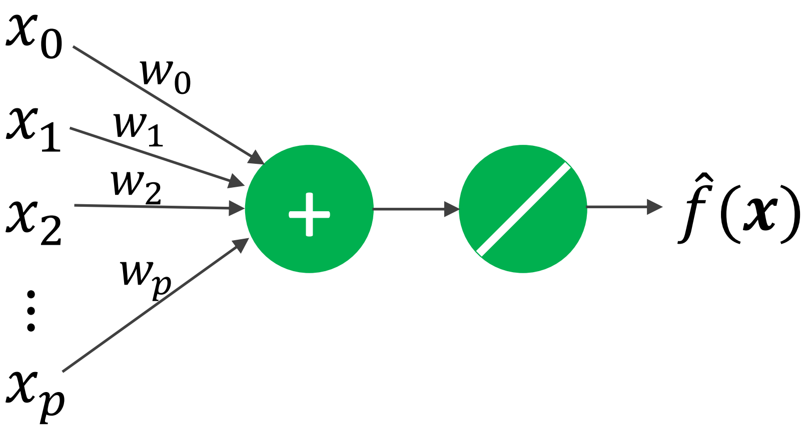
$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$$



Moving from regression to classification

Linear Regression

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

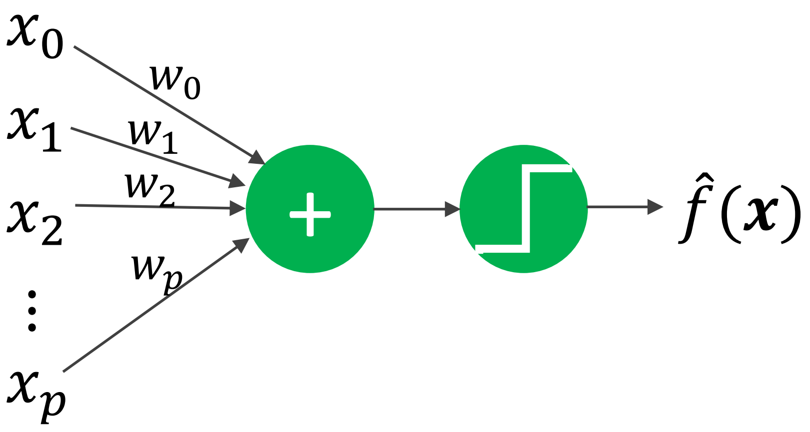


Linear Classification

Perceptron

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

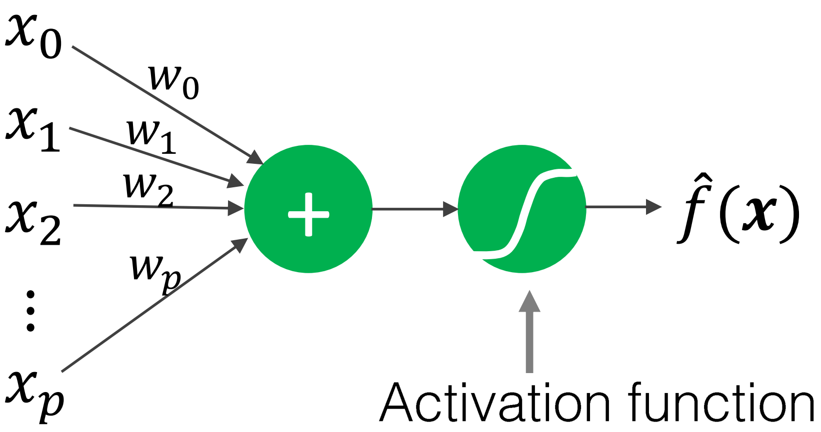
$$sign(x) = \begin{cases} 1 & x > 0 \\ -1 & \text{else} \end{cases}$$



Logistic Regression

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

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



Source: Abu-Mostafa, Learning from Data, Caltech

We fit our model to training data

1. Choose a **hypothesis set of models** to train
2. Identify a **cost function** to measure the model fit to the training data
3. **Optimize** model **parameters** to minimize cost

For linear regression the steps were (i.e. OLS):

a. Calculate the gradient of the cost function
b. Set the gradient to zero
c. Solve for the model parameters

When this approach doesn't work, we typically use **gradient descent**

For classification we COULD try the same cost function as regression

Assume the cost function is mean square error

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

Plug in our model

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

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

Calculate the gradient

$$\nabla_w C(\boldsymbol{w}) = \frac{2}{N}\sum_{n=1}^{N}[\sigma(\boldsymbol{w}^T \boldsymbol{x}_n) - y_n]\sigma(\boldsymbol{w}^T \boldsymbol{x}_n)[\mathbf{1} - \sigma(\boldsymbol{w}^T \boldsymbol{x}_n)]\boldsymbol{x}_n$$

Set the gradient to zero and solve for $\boldsymbol{w}$

$$\nabla_w C(\boldsymbol{w}) = \mathbf{0}$$

But does MSE make sense in this situation?

But we don’t for logistic regression...

Is there a better cost function could we use for classification problems…?

Sidebar: Maximum Likelihood Estimation

We want to determine the underlying probability of the coin landing on “heads” and the coin could be biased.

We flip the coin 1,000 times

…in other words, we have $N = 1{,}000$ **independent** Bernoulli trials

Coin flips, binary outcomes

$$P(X=1) = p$$
$$P(X=0) = 1-p$$

Goal: find the value of p that maximizes the likelihood of our data

Goal: find the value of p that maximizes the likelihood of our data

$$P(X=1)=p$$
$$P(X=0)=1-p$$

For a **single observation**, the likelihood is:

$$L(p)=P(x_i|p)=p^{x_i}(1-p)^{1-x_i}$$

For a **multiple independent observations**, the likelihood is:

For independent random events, the probability of both events is the product of their individual probabilities:
$P(A \text{ and } B)=P(A)P(B)$

$$L(p)=P(\boldsymbol{x}|p)=\prod_{i=1}^{N}P(x_i|p)$$

$$=p^{\sum_{i=1}^{N}x_i}(1-p)^{N-\sum_{i=1}^{N}x_i}$$

Goal: find the value of p that maximizes the likelihood of our data

$$L(p) = p^{\sum x_i}(1-p)^{N-\sum x_i}$$

Maximizing the likelihood is equivalent to maximizing the log-likelihood

$$\ln[L(p)] = \ln[p^{\sum x_i}(1-p)^{N-\sum x_i}]$$

$$\ln[L(p)] = \ln(p)\sum_{i=1}^{N} x_i + \ln(1-p)\left[N - \sum_{i=1}^{N} x_i\right]$$

To **maximize the likelihood**, we take the **derivative of this log likelihood and set it to zero**, then solve for p

Goal: find the value of p that maximizes the likelihood of our data

We take the derivative of this log likelihood and set it to zero, then solve for p

$$\ln[L(p)] = \ln(p) \sum_{i=1}^{N} x_i + \ln(1-p) \left[N - \sum_{i=1}^{N} x_i \right]$$

$$\frac{\partial \ln[L(p)]}{\partial p} = \frac{\sum_{i=1}^{N} x_i}{p} - \frac{N - \sum_{i=1}^{N} x_i}{1-p} = 0$$

This results in our estimate being the mean of our observations:

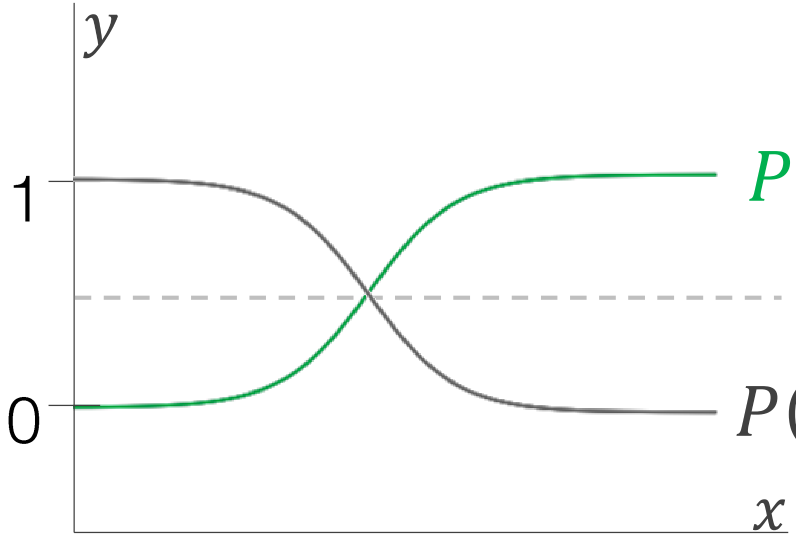
$$\hat{p} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Another interpretation of logistic regression

Our model: $\hat{y} = \hat{f}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$

$$\sigma(\boldsymbol{w}^T\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^T\boldsymbol{x}}}$$

Logistic regression models the **probability that a sample belongs to a class**



$P(y_i = 1|\boldsymbol{x}_i) = \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)$

$P(y_i = 0|\boldsymbol{x}_i) = 1 - \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)$

The interpretation of the Likelihood

The probability of observing the class labels $y_1, y_2, \dots, y_N$ corresponding to $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$

The likelihood for **one observation**:

$$P(y_i|\boldsymbol{x}_i) = P(y_i = 1|\boldsymbol{x}_i)^{y_i} P(y_i = 0|\boldsymbol{x}_i)^{1-y_i}$$

We're interested in the likelihood of the model as a function of the model parameters, $\boldsymbol{w}$. So $P(y_i|\boldsymbol{x}_i)$ is a function of $\boldsymbol{w}$ (see slide 20).

$$L(\boldsymbol{w}) \triangleq P(\boldsymbol{y}|\boldsymbol{X})$$

The likelihood for **all observations**:

$$P(\boldsymbol{y}|\boldsymbol{X}) = P(y_1, y_2, \dots, y_N|\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N) = \prod_{i=1}^{N} P(y_i|\boldsymbol{x}_i)$$

Source: Malik Magdon-Ismail, Learning from Data

The likelihood for all observations:

$$P(\boldsymbol{y}|\boldsymbol{X}) = \prod_{i=1}^{N} P(y_i|\boldsymbol{x}_i) = \prod_{i=1}^{N} P(y_i = 1|\boldsymbol{x}_i)^{y_i} P(y_i = 0|\boldsymbol{x}_i)^{1-y_i}$$

Substituting:
$$P(y_i = 1|\boldsymbol{x}_i) = \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)$$
$$P(y_i = 0|\boldsymbol{x}_i) = 1 - \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)$$

$$= \prod_{i=1}^{N} \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)^{y_i}[1 - \sigma(\boldsymbol{w}^T\boldsymbol{x}_i)]^{1-y_i}$$

We want to **MAXIMIZE the likelihood (minimize it's negative)**

We can take the **logarithm**, negate it to get our **cost function**, then minimize it (using the gradient)

A little algebra

$$P(\boldsymbol{y}|\boldsymbol{X}) = \prod_{i=1}^{N} \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)^{y_i}[1-\sigma(\boldsymbol{w}^T \boldsymbol{x}_i)]^{1-y_i}$$

$$= \prod_{i=1}^{N} \hat{y}_i^{\,y_i}[1-\hat{y}_i]^{1-y_i}$$

assuming $\hat{y}_i \triangleq \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)$

If we take the log of both sides:

$$\log P(\boldsymbol{y}|\boldsymbol{X}) = \log\left[\prod_{i=1}^{N} \hat{y}_i^{\,y_i}[1-\hat{y}_i]^{1-y_i}\right] = \sum_{i=1}^{N} \log(\hat{y}_i^{\,y_i}[1-\hat{y}_i]^{1-y_i})$$

$$= \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$$

Recall that
$\log(ab) = \log(a) + \log(b)$

$$\log P(\boldsymbol{y}|\boldsymbol{X}) = \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$

We can define our cost function: $C(\boldsymbol{w}) = -\log P(\boldsymbol{y}|\boldsymbol{X})$

$$\boxed{C(\boldsymbol{w}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)}$$

For logistic regression,

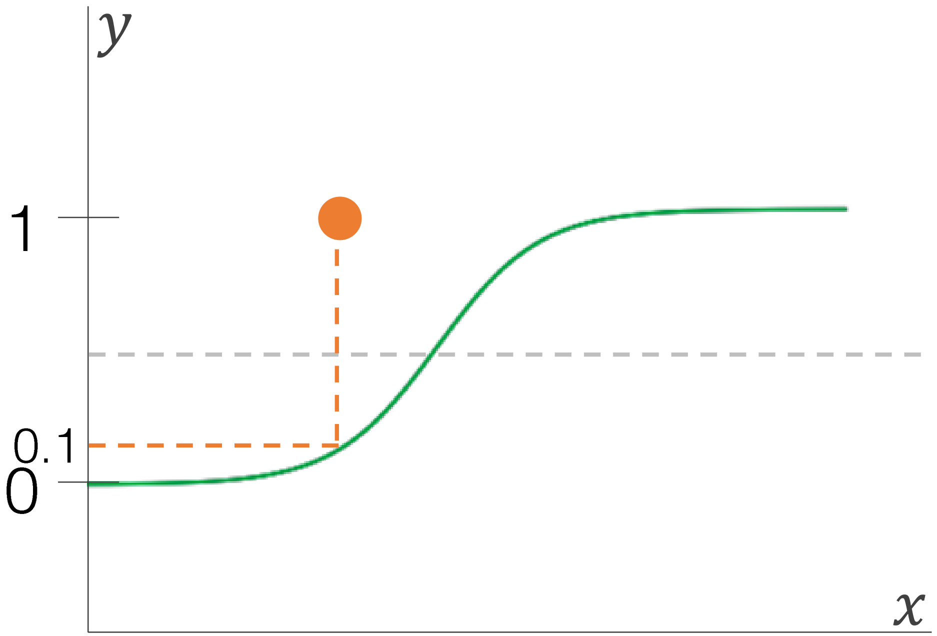
$$\hat{y}_i \triangleq \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)$$

This is the cross entropy cost function

Mean Square Error vs Cross Entropy

$$\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$



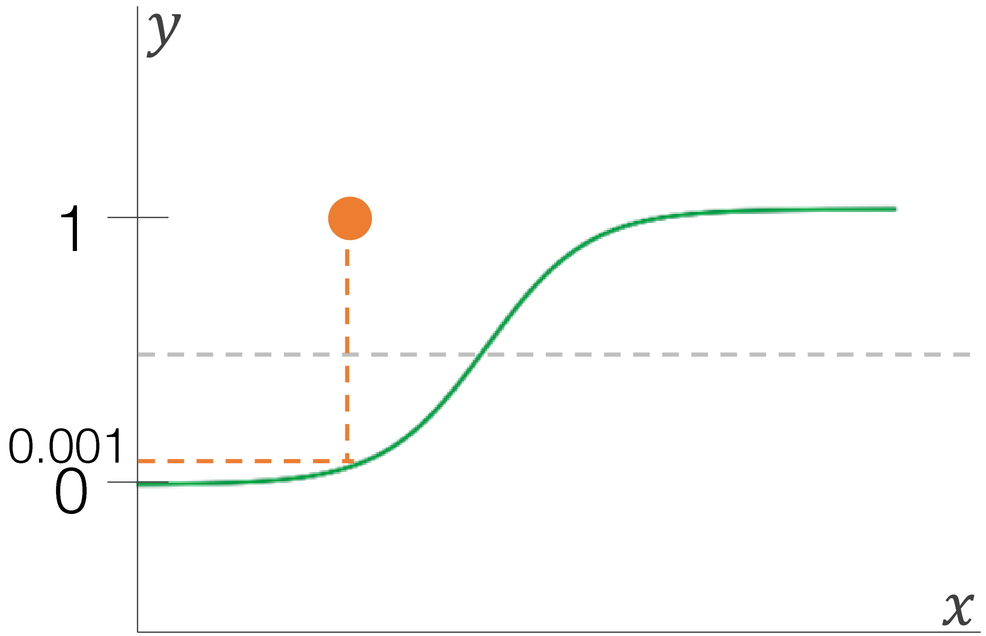
$$\begin{aligned} C_{MSE} &= (\hat{y}_i - y_i)^2 \\ &= (0.1 - 1)^2 \\ &= 0.81 \end{aligned}$$

$$\begin{aligned} C_{CE} &= -[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \\ &= -[(1)\log(0.1) + (0)\log(0.9)] \\ &= 2.30 \end{aligned}$$

Mean Square Error vs Cross Entropy

$$\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$



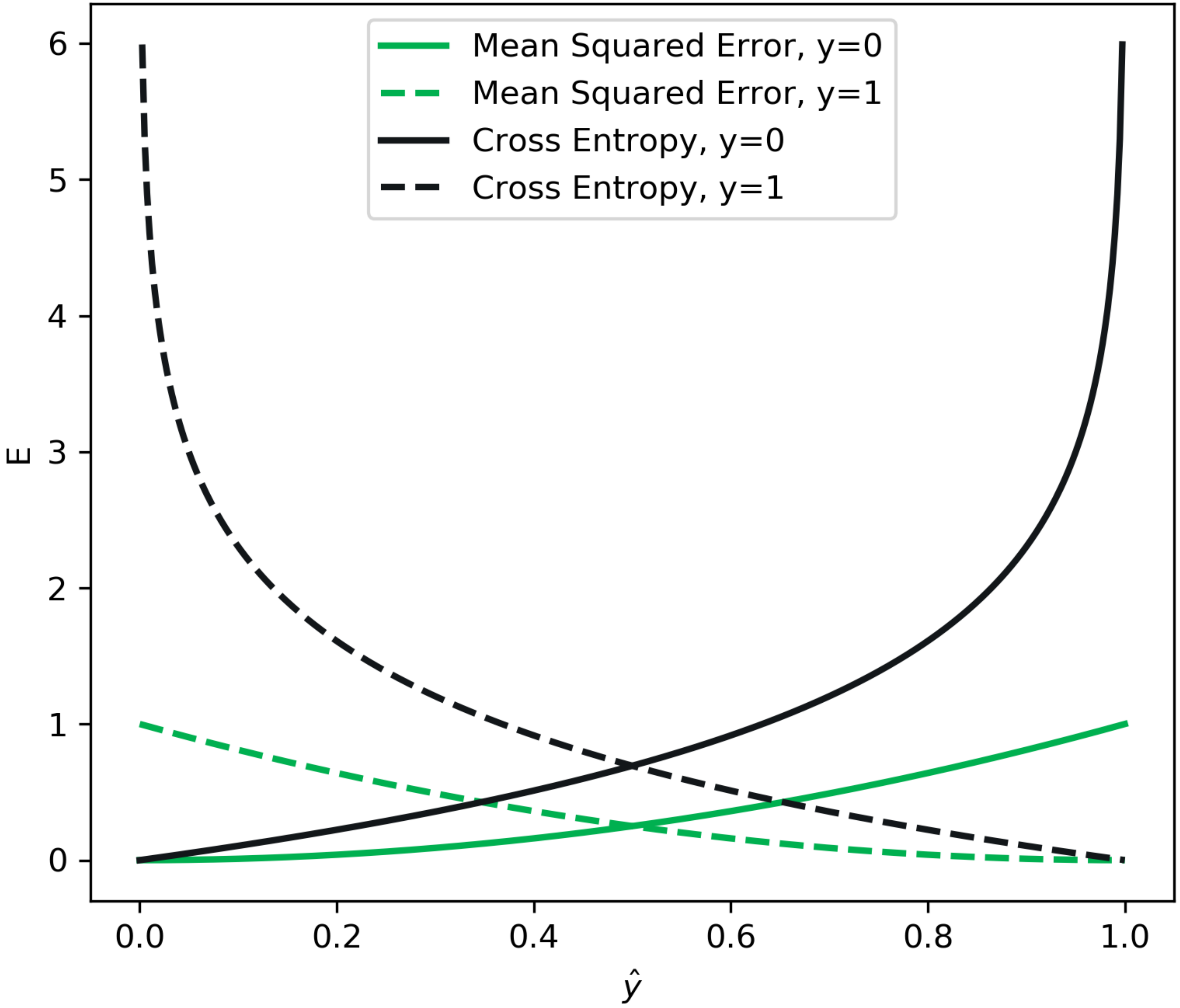
$$\begin{aligned}
C_{MSE} &= (\hat{y}_i - y_i)^2 \\
&= (0.001 - 1)^2 \\
&= 0.998
\end{aligned}$$

$$\begin{aligned}
C_{CE} &= -[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \\
&= -[(1)\log(0.001) + (0)\log(0.999)] \\
&= 6.91
\end{aligned}$$

Cross Entropy vs MSE

If a model is wrong, but is highly confident, it faces exponentially larger penalties with cross-entropy

Cross-entropy as a loss function converges more quickly than MSE for classification when fitting the model



Logistic regression does not have a closed-form solution like linear regression did

We need a new approach…

Gradient descent

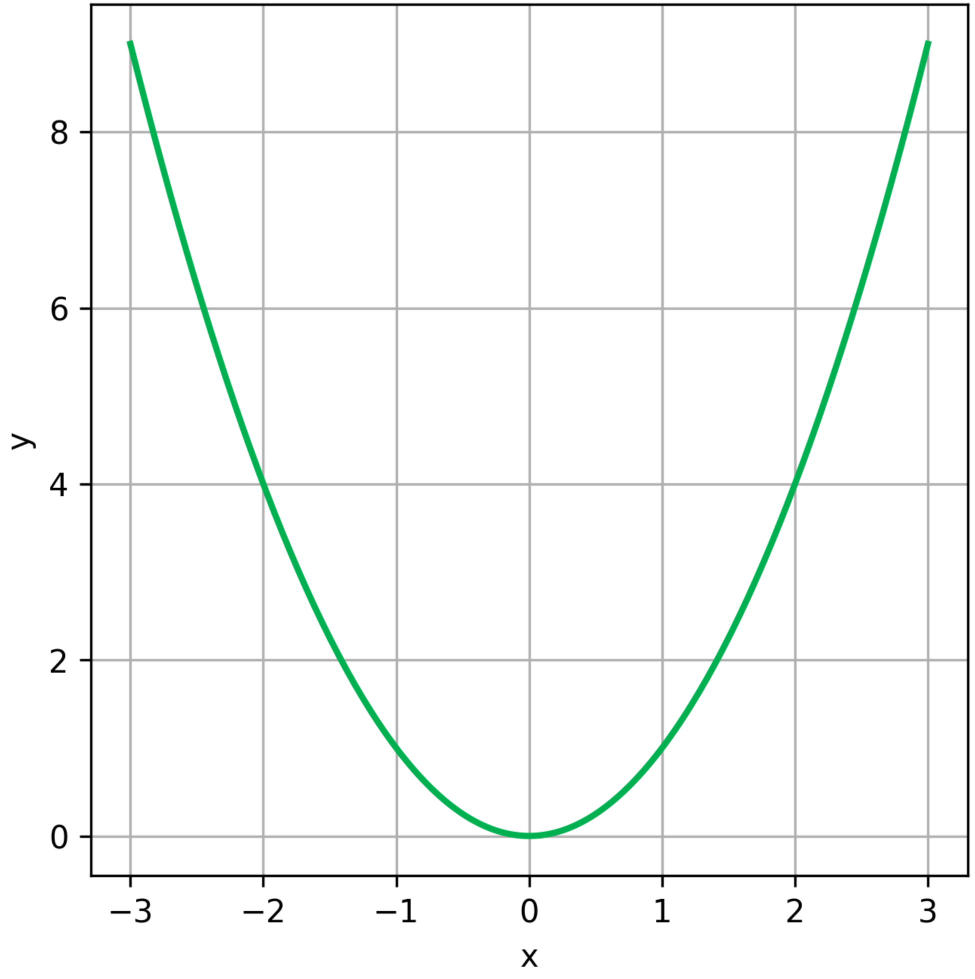
Minimize $y = x^2$

We start at an initial point and want to "roll" down to the minimum

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \eta \boldsymbol{v}$$

Learning rate

Direction to move in



Gradient descent

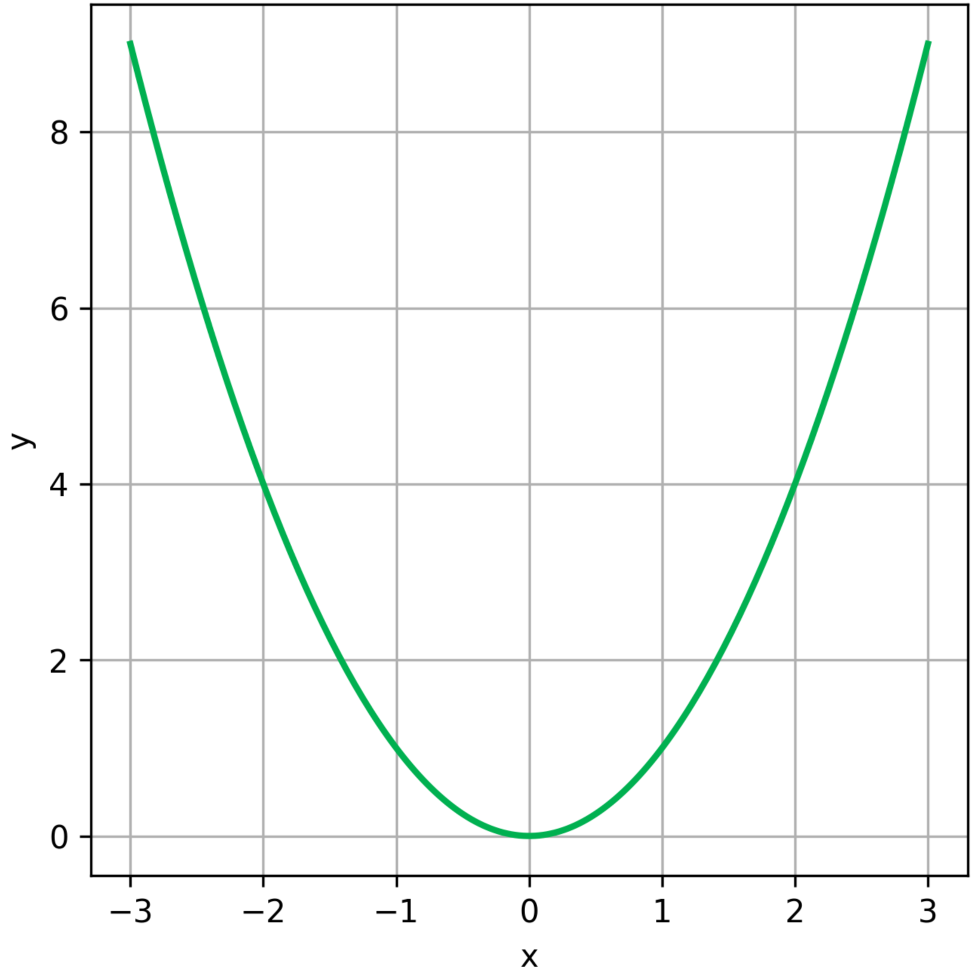
Minimize $f(x) = x^2$

The gradient points in the direction of steepest **positive** change

$$\frac{df(x)}{dx} = 2x$$

We want to move in the **opposite** direction of the gradient

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - \eta \nabla f\left(\boldsymbol{x}^{(i)}\right)$$



Gradient descent

Derivative:

$$\frac{df(x)}{dx} = 2x$$

Gradient descent update equation:

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - \eta \nabla f\left(\boldsymbol{x}^{(i)}\right)$$

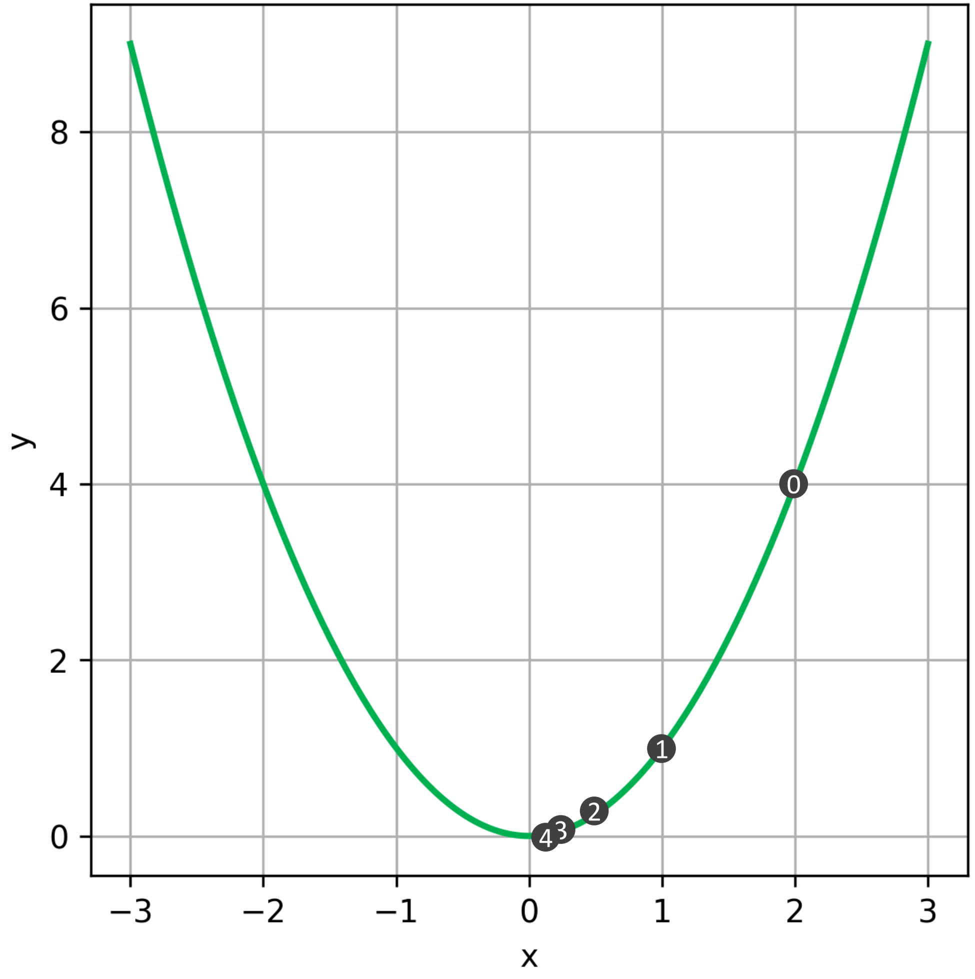
Minimize $f(x) = x^2$

Assume $x^{(0)} = 2$ and $\eta = 0.25$

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - (0.25)(2\boldsymbol{x}^{(i)})$$

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - (0.5)\boldsymbol{x}^{(i)}$$

i	$x^{(i)}$	$y^{(i)}$
0	2	4
1	1	1
2	0.5	0.25
3	0.25	0.0625
4	0.125	0.0156



Takeaways

Transformations of features (**feature extraction**) may help to overcome nonlinearities

Logistic regression is much better suited for classification than linear regression

Logistic regression parameters must be estimated iteratively, and a method for that optimization is **gradient descent**

Gradient descent can be used for **cost function optimization** and there are a number of variants